10/088258
JC13 Rec'd PCT/PTO 13 MAR 2002

EXPRESS MAIL NO. EL652175896US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Siani Lynne PEARSON, et al.	Re: Claim To Priority
U.S. Appln. No.: not yet assigned	Group: not yet assigned
U.S. Filing Date: concurrently herewith	Examiner: not yet assigned
International Application No: PCT/GB00/03689	
International Filing Date: 25 September 2000	Our Ref.: B-4528PCT 619575-6
For: "TRUSTED PLATFORM FOR RESTRICTING USE OF DATA"	Date: March 13, 2002

Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Attn: United States Designated/Elected Office (DO/EO/US)

35 U.S.C. 119 CLAIM TO PRIORITY

Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Attn: United States Designated/Elected Office (DO/EO/US)

Sir:

Prior PCT International Application No. PCT/GB00/03689, designating the U.S., claims foreign priority as follows:

COUNTRY	FILING DATE	SERIAL NUMBER
GREAT BRITAIN	25 September 1999	9922665.6

A certified copy has been filed in prior PCT International Patent Application No. PCT/GB00/03689.

Applicants hereby confirm that this claim for priority applies to the above-identified U.S. International Stage Application.

Respectfully submitted,

Richard P. Berg
Reg. No. 28,145
Attorney for Applicant
LADAS & PARRY
5670 Wilshire Boulevard #2100
Los Angeles, California 90036
(323) 934-2300

PCT/GB 00/03689

The Patent Office

REC'D 06 OCT [illegible]

WIPO PCT

INVESTOR IN PEOPLE

#2

10/088258

GB00/03689

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

EJU

PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 22 NOV 1999

Patents Form 1/77

Patents Act 1977
(Rule 16)

THE PATENT OFFICE
25 SEP 1999
RECEIVED BY POST

The Patent Office

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference — 30990134 GB

2. Patent application number *(The Patent Office will fill in this part)* — **9922665.6**

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

   Hewlett-Packard Company
   3000 Hanover Street
   Palo Alto
   California 94304 USA

   Patents ADP number *(if you know it)* — 06293385001

   If the applicant is a corporate body, give the country/state of its incorporation — Delaware USA

4. Title of the invention

   A METHOD OF ENFORCING TRUSTED FUNCTIONALITY IN A FULL FUNCTION PLATFORM

5. Name of your agent *(if you have one)* — LAWMAN Matthew John Mitchell

   "Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

   Hewlett-Packard Limited
   IP Section
   Filton Road
   Stoke Gifford
   BRISTOL BS34 8QZ

   Patents ADP number *(if you know it)* — 07337009001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:* — Yes

   a) any applicant named in part 3 is not an inventor, or
   b) there is an inventor who is not named as an applicant, or
   c) any named applicant is a corporate body.
   See note (d))

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	
Description	11
Claim(s)	
Abstract	
Drawing(s)	2 + 2

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	X
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	Annex 'A' - unpublished European patent application No 99301100.6 and Annex 'B' ~~unpublished European patent application~~ No 99304164.9

11. I/We request the grant of a patent on the basis of this application.

Signature Matthew Lawman   Date 24/09/99

12. Name and daytime telephone number of person to contact in the United Kingdom   Katerina Nommeots-Nomm   0117 312 9947

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

f) *For details of the fee and ways to pay please contact the Patent Office.*

A method of enforcing trusted terminal functionality in a full function platform

(HP Ref: 30990134)

This invention concerns the provision of a method for enforcing trusted terminal functionality in a full function platform; this enables the display of data that was processed remotely, while preventing the misuse of that data. Benefits can be obtained for client, server, or developer, as the system can be used for a wide range of services, including protecting private information, licensing data or allowing full functionality trial software. The client platform can be trusted to output data faithfully, in such a way that the data itself cannot be copied or modified. Hence full functionality can be used for trial software, which is rarely the case at present because of the security risks involved. End-users can also gain through sensitive information such as e-mail messages not being stored on the hard disk of the client machine, and hence in hot-desking situations such as using a shared terminal in a public place, such information being less liable to attacks on its confidentiality or integrity.

In this specification, 'data' signifies anything that can be formatted digitally, such as images, software and streaming media.

In the future, computer systems will be able to achieve a more secure booting, together with integrity checks on other code to ensure that viruses or other unauthorised modifications have not been made to the operating systems and mounted software. In addition, a new generation of tamper-proof devices are already appearing or will soon appear on the market and include both external or portable modules (such as smart cards) and internal modules (embedded processors, semi-embedded processors or co-processors with security functionality, i.e. including motherboard, USB (Universal Serial Bus) and ISA (Industry Standard Architecture) implementations). These tamper-proof modules will be used to check that the hardware of the system has not been tampered with, and to provide a more reliable form of machine identity than currently available (for example, the Ethernet name). Yet how to counteract piracy, and how to licence and meter software in a manner that is acceptable to software developers and end-users will still be a very important problem.

Software licensing is subject to hackers and piracy, and all the current software licensing methods used have problems associated with them. Software implementations of licensing (such as "licence management systems") are flexible, but not especially secure or fast. In particular, they suffer from a lack of security (for example, being subject to a generic "hack") and difficulty in genuine replacement of software. Conversely, hardware implementations ("dongles") are faster and generally more secure than software

implementations, but inflexible. They are tailored only for a particular piece of software and are inconvenient for end-users.

The present invention, in its preferred embodiment, seeks to deliver the best of both worlds: a hardware implementation that is secure and fast, but with the convenience and flexibility of a software implementation.

Prior art in the field of content protection includes techniques such as watermarking of content, software wrappers around content, protecting passwords and fingerprinting techniques. In addition, there are various approaches that involve encryption of content that is sent to the client machine, and a decryption key being sent to the client machine in order that it can decrypt the content. All these approaches suffer from the potential drawback that the client machine could be untrustworthy, and that the data could be abused once decrypted or otherwise made available to the client machine (for example, by the protection mechanism being hacked, or by the clear version being copied). The approach described in this patent differs in that it is not based on any such working model: instead, at least part of the information is temporarily stored in protected memory either within or only accessible by the tamper-resistant hardware before deletion, and this part is not stored on the hard disk. The tamper-resistant hardware is used for authentication, for controlling the output of the image and optionally for billing. The client machine never gets the whole data package (protected or unprotected), as it does in the models described previously, and so it is not possible to abuse the software via the client machine in that manner. Hence, for example, the user will be able to copy the screen or retype text from the screen, but will not be able to copy an original document; in the case of music, the user will be able to listen to a soundtrack and record the sound in the room, but will not be able to copy the digital object directly. This cuts down the attractiveness of piracy a great deal.

In addition to the benefits of protection against copying and unauthorised use of data, and increased flexibility in licensing models such as pay-per-use and time-dependent models, the invention offers protection against hacking attempts such as modification or deletion of data wrappers stored on the client platform, since such storage never takes place in this model and the tamper-resistant hardware within the client platform protects against alteration of any image within the platform. More specifically, if data access is allowed to users on a trial basis, at present the danger of copying or modification of the usage controls upon such data is generally considered too large to allow anything but an inferior product to

be sent for trial usage. The system described in this patent allows software with full functionality, or images with full resolution, to be examined by end-users.

Although the system can be used for the purposes of software licensing, or provision of full functionality trial software as mentioned above, it can be used instead or in conjunction with these in order to also protect private information of the client. For example, if an end-user logs in to a shared terminal containing tamper-resistant hardware in order to access private information, possibly using remote login, then this information is only stored in protected memory either within or accessible only via the hardware and not on the hard disk, and can be deleted entirely after the user has logged out.

Applicants prior patent application EP 99301100.6, the entire contents of which are hereby incorporated herein by reference (a copy of which accompanies this application as Annex A), describes the use of a Trusted Component to enable verification of the integrity of a computer platform by the reliable measurement and reliable reporting of integrity metrics. This enables the verification of the integrity of a platform by either a local user or a remote entity. That prior patent application describes a general method of reporting integrity metrics and verifying the correctness of the integrity of a platform by comparing reported values of metrics with proper values of metrics.

A further one of applicant's patent applications EP 99304164.9, the entire contents of which are hereby incorporated herein by reference (a copy of which accompanies this application as Annex B), describes a method of displaying images sent to the trusted platform such that the image appearing on the monitor can be trusted to correspond to the data input to the client. The present invention uses two trusted computer platforms to allow trusted display of data that has been processed remotely, where the platforms' integrity is reported using the method of the first prior patent application and where one sends an image to the other which is then displayed in the manner described in the second patent application, and at least part of which is stored in protected memory within or only accessible by the Trusted Component.

In overview, the embodiment of the present invention uses a tamper-proof component, or "trusted module" of a computer platform in conjunction with software, preferably running within the tamper-proof component, that controls manipulation of and selections relating to a data image to be transferred between two such computer platforms. Thereby, trusted terminal functionality is provided in a full function platform. Metering records

can be stored in a tamper-proof device or smart card and reported back to administrators as required. There can be an associated clearinghouse mechanism to enable registration and payment for data.

More formally, in accordance with a first aspect of the present invention, the invention consists of a licensing system comprising at least two computer platforms, one acting as server and one as client, which are connected by a secure communications path. Each computer platform has: a trusted module which is resistant to internal tampering and which stores a third party's public key certificate; means of storing remote imaging code (in the case of the server, remote image sending code for providing an interface for sending information from the server to other trusted platforms corresponding to an image of data executing upon the server; in the case of the client, remote image receiving code for providing an interface for receiving information from other trusted platforms corresponding to an image of data which may be displayed upon the monitor of the client platform and/or capturing user selections relating to the running of such an image and relaying these back to the server platform); and means of storing a hashed version of the remote imaging code signed with the third party's private key; wherein the computer platform is programmed so that, upon booting of the platform: the remote imaging code is integrity checked with reference to the signed version and the public key certificate; and if the integrity check fails, the remote imaging code is prevented from being loaded. If the integrity check fails, it may be arranged that the complete platform integrity fails. One or more smart cards, with an associated reader, are an additional, optional part of the computer platform.

The trusted module or component, as described in EP 99301100.6, is preferably immune to unauthorised modification or inspection of internal data. It is physical to prevent forgery, tamper-resistant to prevent counterfeiting, and preferably has crypto functions to securely communicate at a distance. Methods of building trusted modules are, per se, well known to those skilled in the art. The trusted module may use cryptographic methods to give itself a cryptographic identity and to provide authenticity, integrity, confidentiality, guard against replay attacks, make digital signatures, and use digital certificates as required. These and other crypto methods and their initialisation are well known to those skilled in the art of security.

Optionally, part of the functionality of the remote imaging code may be carried out by hardware within the local trusted component rather than software.

Next the way in which these components interact to form a system for trusted terminal functionality in a full function platform will be described. The extreme form of the general model is that licensed data is executed on the server, and not on the client. In return for payment, the client receives imaging information corresponding to the execution of the data on the trusted server. This is sent via the remote image sending code on the server. Thereafter, the remote image receiving code on the client machine sends to the server keyboard strokes, corresponding to the selections of the user, and receives in return imaging information, corresponding to the changing execution of the application. The imaging information is sent directly from the trusted server via a secure channel such as PPTP to the TC within the client, which displays the imaging information directly without having to involve any untrusted parts of the computing apparatus (cf. previous patent).

There are various different types of approach regarding how much software actually runs on the client. It is not efficient in all cases to run software on the server rather than the client. For more sensitive information, such as data access or where the information may contain a great deal of overlap each time the application runs, it might be applicable to temporarily store all of the image in the protected memory, and have the software displayed on the client, but actually running on the server. The client at no stage stores the software apart from in the protected memory, and therefore is not liable to licence infringement attacks on the data via the hard disk or other storage media. For less sensitive information, and especially where an application may produce differing images each time it is run, as is usually the case with game software, it would probably be more appropriate to run the software only partly from the server; for example, locally, yet needing input from the server such as an on-line service in order to run the software. The server must still have overall control, so that although the client machine may be able to run the program, this cannot happen without the server being involved. There are different ways of carrying this out: for example, the server could supply key bits of the information, and transmit the image in communal blocks which would be the same for all clients, with the client TC repeatedly authenticating to the server TC for personalised information or key bits; or some of the data could be stored locally, with the server transmitting additional data to the protected memory. For the sake of efficiency, during and after execution, only part of the information (such as the key bits) is stored in the protected memory, and the rest can be stored on the hard disk or other storage media. This partial model of image transfer can be used concurrently with total models for different data on the same server.

The server is in a trusted environment, which is protected against data and wrappers being altered or copied. Hence, licensing models such as pay-per-use and time-dependent models, as well as more traditional models, may be used in a secure manner.

Preferably, as described in applicants co-pending patent application EP 99304164.9, there can be a trusted display module within the TC itself to increase the trust residing in the image.

Preferably, to increase security, as described in applicants co-pending patent application EP 99304164.9, a seal image can be displayed on the client screen which only the owner of the smart card inserted into the reader knows to be their correct seal image, in order to check the security of the connection between the client and server. Before the smart card owner carries out sensitive tasks such as providing billing information, the smart card requires authentication from the TC of the client platform that is enhanced by the seal image being shown on the client monitor and requiring authorisation from the smart card owner before any sensitive information is conveyed.

As an option, the display of a selected area of pixels on the trusted client platform may be reserved for an alternative usage by the server trusted platform, perhaps on behalf of a third party. The given pixel area can vary over time, and convey information that may not directly be related to the data executing on the trusted server platform. This allows adverts or other proprietary information to be incorporated into the display image sent by the server trusted platform, or a trusted third party.

An embodiment of the present invention will now be described with reference to the accompanying drawings.

Figure 1 illustrates the physical system of the computer platforms. The Trusted Component [103] is in the path between normal computer host [101] and the display [104]. This enables the TC [103] to reliably write to the display, without fear of subversion from normal software, including the operating system. The host computer is connected to a keyboard [105] that has a built-in smart card reader. A smart card [107] is plugged into the keyboard and can be accessed by the host system [101] and the TC [103]. The smart card [107] is able to communicate securely with the TC [103]. (It should be noted that this architecture is that described in EP 99304164.9 and is not part of this invention.)

Figure 2 illustrates the basic licensing system, in which an end-user logging into a client trusted platform [101] (including a TC [103] and possibly a smart card reader [107]) may access data [207] which is running on a server trusted platform [209] (including a TC [206]). Preferably, the protected memory and remote image receiving code are located within the TC [103], and hence the secure communications path [202] is unnecessary. The data image [208] is sent to the client TC [103] via a secure path such as PPTP and then on directly to the display [104]. Input devices such as the keyboard [105] capture user input choices and this information is sent to the client TC [103] and relayed across to the server TC [209] via a secure path. Such communication is repeated whenever appropriate updates of display corresponding to data execution or user inputs need to be made. Various models are possible, in which an ISP runs the trusted server, or acts as a middle-man for a developer running the trusted server. Let C be the entity controlling the running of the data, and hence the display. C could be the developer controlling the trusted server platform or a third party who is trusted by the developer to either execute the data on their own trusted platform, or to relay the image obtained from the developer. C has full control over the display, and hence can 'reserve' certain areas of pixels [203] for proprietary information, advertisements, or other information not directly associated with the trial software itself. This information could be static or change with time, or consist of a streaming media such as video. The areas of pixels could change themselves with time, and may or may not cover the executing trial data, depending upon C's chosen implementation. As an alternative, the whole screen could be taken over for advertisements for given time intervals. In the case of audio data, analogously, a break could be made at certain time intervals for pre-recorded advertisements.

Preferably, there is a dedicated communications link [202] between the client TC [103] and the client protected memory [201] accessible only to the client TC [103].

The client TC [103] must be able to carry out the following functionalities:

Controlling output of image corresponding to execution of data

Authentication of server TC

The following additional optional functionalities are preferable:

verification of data protection capabilities of server TC

authentication of user's smart card

billing, reporting or metering capabilities

protected memory within TC (although TC storage capability would have to be very large to do this)

The server TC [209] must be able to carry out the following functionalities:

Authentication of client TC or smart card

Protection of client's sensitive information (obtained via registration or metering)

Optionally, billing, reporting or metering capabilities

The user's smart card [107], if used in the model, needs to be able to authenticate to the client TC or server TC, depending upon the trust model, and possibly to verify the data protection capabilities of the server TC.

The following is an example of a procedure by which a system for trusted terminal functionality in a full function platform, denoted 'B', is provided. The kind of situation when this might be used is where the user of B wishes to view a document or access useful information, but would have to pay more in order to copy the original document.

Initial Setup

The end-user registers the client TC of B or his smart card, as appropriate for the desired payment or licensing model, with the trusted platform A, acting as server.

The user may arrange to pay for the data at this stage, or the TC of B or his smart card would be charged up in advance, and the data purchase recorded on this device and reported back to A at a later date (see discussion below).

A already has the public key of B or the smart card via the original registration process; if not, this is sent to A (cf. stage 4).

The public key certificate for A is installed by A into the TC of B, or smart card (as appropriate). A suitable protocol which would incorporate authentication from A to the TC would be that, in response to a request for authentication from the TC incorporating a nonce generated by the TC, A returns a message which includes its public key certificate and the nonce, signed with its private key. The TC can then check that the message came from A.

Preferably, the user checks that the server A can be trusted, by his smart card or B's TC verifying the protection capabilities of A's TC.

Data Execution

The user requests the data to be displayed, via the OS on B.

There is mutual authentication between the TC of A and the TC of B. If a smart card is used in the registration process, there could be mutual authentication between the smart card and the TC of A, and/or between the smart card and the TC of B.

Optionally, if the authentication between the smart card and the TC of B is successful, the user will see a special displayed message, which can be a number, a letter, a colour or a picture which comes from the smart card and is not known to B before the mutual

authentication protocol. The user will be asked to give confirmation that he wishes the protocol to continue, and leave the smart card in the reader of B (the exact type of confirmation depends upon the trusted relationship between the user and client machine). He may be asked to input a password.

The image corresponding to execution of data on the trusted platform A is transferred from that platform via a secure channel such as PPTP to the protected memory within the trusted platform B

Optionally, an image transfer log is made (see below)

Optionally, B performs an integrity check on the image by checking the signature using A's public key, and verifying whether it is from the expected source

That image is then displayed on the trusted platform B directly via its trusted component (for example, using the methods described in the trusted interface patent) with at least part of this image stored within the protected memory of B

Input device information corresponding to the user's selections on B is returned to A, preferably via the secure communications channel

Execution of the data on A is modified corresponding to this selection, if appropriate

Stages 9,12 and 13 above are repeated to represent any appropriate changing display states corresponding to the data execution

The user or server may initiate a request to end the session

Optionally, a usage log is made (see below) and/or a billing charge made to the user.

Registration using a smart card rather than the client machine's TC potentially allows the user to access data paid for and licensed to his smart card across any secure client platform.

Preferably, the communications between A and B will be protected with the security feature of confidentiality. There are many different possible authentication or key generation methods, and the most applicable one is dependent upon the trusted relationship between SC and B's TC. In these methods, A and B will establish a session key used to encrypt data transmitted between A and B. Checks for authenticity and integrity on messages between A and B may be incorporated into such a protocol, using means well known to those skilled in the art.

If the data execution is not free, there are at least three options for billing for the service provided by the trusted server platform.

The first is that billing and usage information is stored on the trusted server platform at the time of registration or initial payment by the trusted client platform, or at the time of image transfer to the client platform. Preferably, such information would be stored within the TC of the trusted server platform, although there may well not be enough storage for this, or, if stored elsewhere, protected using keys stored within the TC.

The second option is for a logfile or other usage information to be stored on the TC of the trusted client platform. The server (preferably, server TC) could either interrogate the TC to find out relevant information, or else the client TC could report this information back to the server TC at convenient intervals. One way of doing the latter would be for the user's TC to download an applet from the server platform to allow reporting back and billing to be made, so that, for example, the server platform can check what data has already been displayed by that user. The server can act as a clearinghouse, with information relating to billing being sent back via the downloaded software within the user's TC. Optionally, payment can be made via credit card details provided by the TC or smart card.

The third option is for a logfile or other usage information to be stored on the user's smart card. This would measure usage by a particular user, rather than on a particular machine, and would be more appropriate for certain environments such as those where hot-desking is common. Again, the server could either interrogate the smart card to find out relevant information, or else the smart card could report this information back to the server TC at convenient intervals, perhaps by means of downloaded applets from the server to the user's smart card in an analogous manner to that described above.

This data usage can be used for billing purposes as well as for checking what data has been accessed by the user. This is of particular relevance in cases where data access should not exceed a certain number of times, such as fixed usage models of licensing or trial software. In the case of trial software, it is necessary to check the logfile stored on the server TC, client TC or smart card corresponding to the user or machine, as desired or appropriate, and check before the trial software is executed that there is not a case of prior usage. If there is evidence from the log that the software has already been tried, then an error message will be generated via the server TC to the client TC, the software will not be executed on the server and instead an error message displayed upon the screen of the trusted client platform to the effect that multiple downloading of this software is not permitted. This check allows the safe usage of full functionality trial software.

Trial software is provided with full functionality for a time period specified by the developer. The software is run on a trusted server, and displayed to the screen of a trusted client, as described above. To prevent serial use of trial software with full functionality, the server needs to record that the machine or user has already used it. The client is unable to copy the software or use it for longer than the allocated period, or contravening the terms of the licensing agreement.

When the user signs a licence agreement, s/he agrees to his smart card or TC being updated with a logfile that the trial software has been downloaded, and to the server being able to check and add to such a logfile in future. Before the software image is sent to the client, a check is made in the logfile whether this trial has been used before, and a decision made by the server whether to continue with the trial. When the software image is sent to the client, a note is made in the logfile that the trial has been used. After a certain time, the user is prompted for payment if s/he requires continued use of the software.

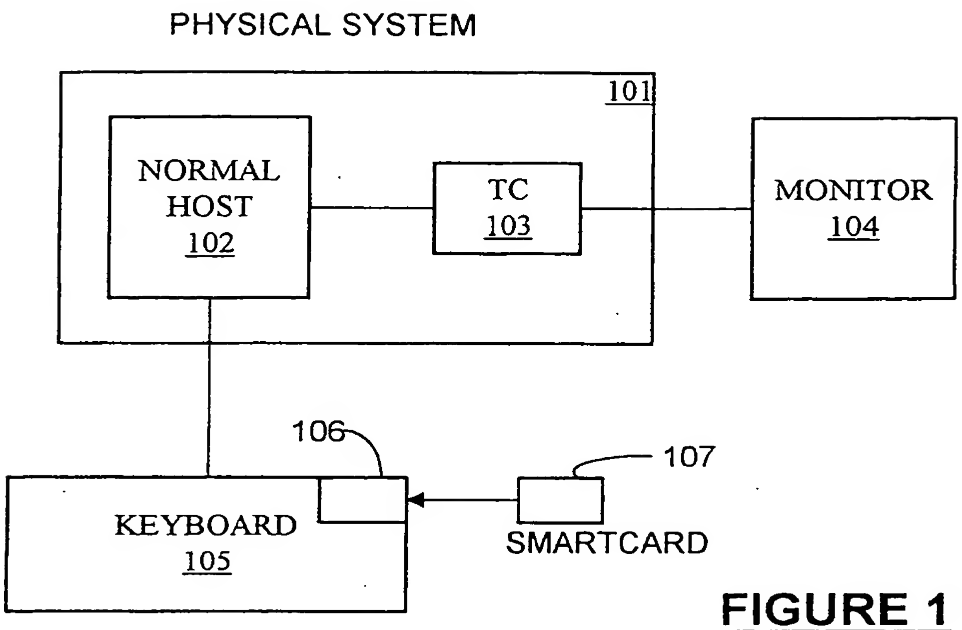
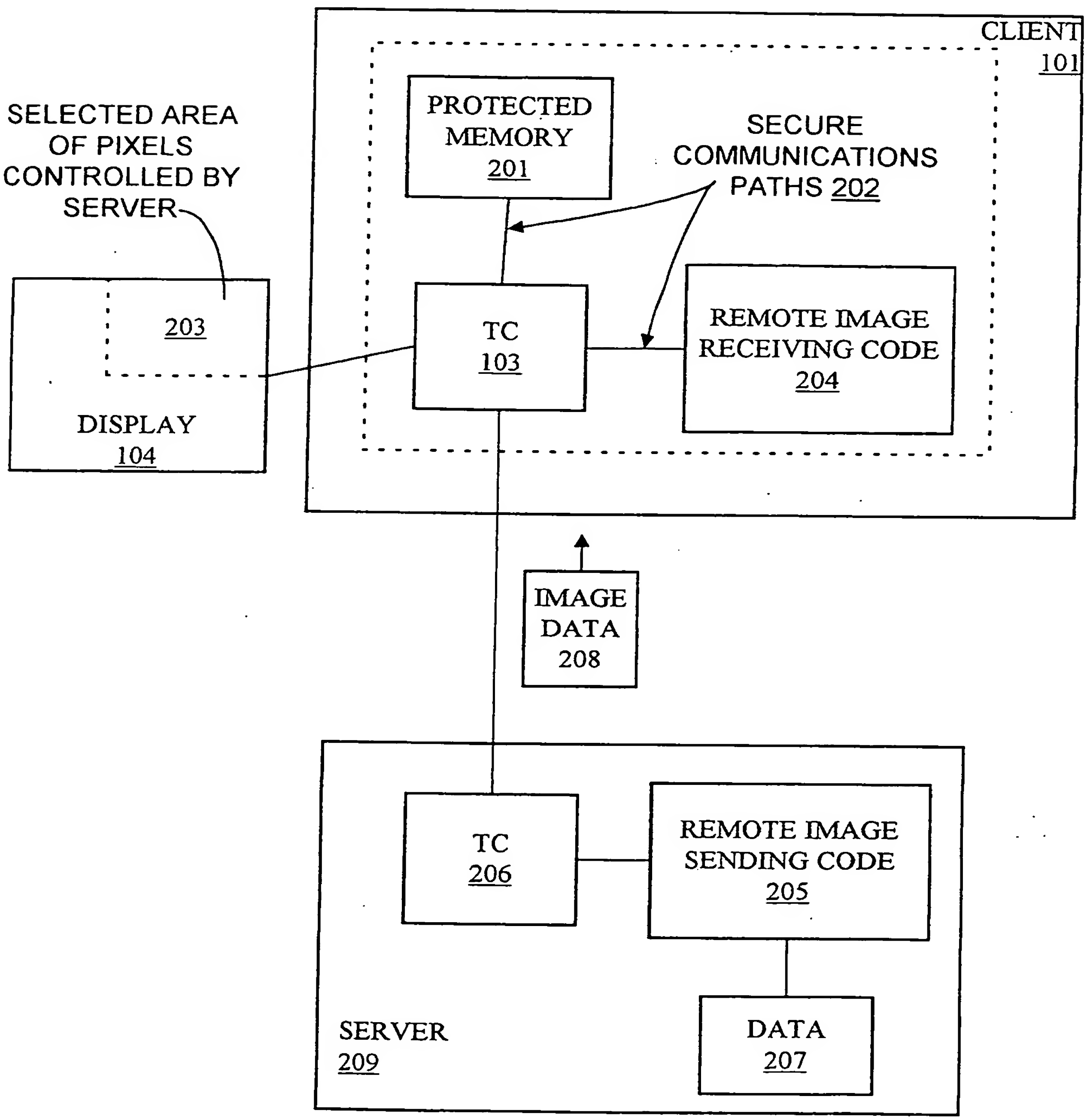

PHYSICAL SYSTEM
101
NORMAL HOST 102
TC 103
MONITOR 104
106
107
KEYBOARD 105
SMARTCARD

FIGURE 1



LOGICAL DIAGRAM OF IMAGE TRANSFER SYSTEM

FIGURE 2

Trusted Computing Platform

(HP Ref 30990050)

Technical Field

The present invention generally relates to trusted devices, trusted computing platforms, trusted transactions and methods of operating the same.

Background Art

For commercial applications, a client computing platform typically operates in an environment where its behaviour is vulnerable to modification by local or remote entities. This potential insecurity of the platform is a limitation on its use by local parties who might otherwise be willing to use the platform, or remote parties who might otherwise communicate with the platform; for example, for the purposes of E-commerce. For the present purposes, both local parties and remote parties will be referred to as "users" unless otherwise stated.

Existing security applications, for example virus detection software, execute on computing platforms under the assumption that the platform will operate as intended and that the platform will not subvert processes and applications. This is a valid assumption provided that the intended software state has not become unstable or has not been damaged by other software such as viruses. Users, therefore, typically restrict the use of such platforms to non-critical applications, and weigh the convenience of the using the platforms against the risk to sensitive or business critical data.

Increasing the level of trust in platforms therefore enables greater user confidence in existing security applications (such as the 'Secure Sockets Layer' or 'IPSec') or remote management applications. This enables greater reliance on those applications and hence reduced 'cost of ownership'. Greater trust also enables new electronic methods of business, since there is greater confidence in the correct operation of both local and remote computing platforms.

In this document, the word 'trust' is used in the sense that something can be 'trusted' if it always behaves in the way it is expected to behave.

Disclosure of the Invention

The present inventors have appreciated that it is desirable to use a physical device in a computing platform to verify and possibly enforce trust in that platform. Typically, the device provides trusted measurement and reporting of attributes of the associated platform, which indicate the integrity of the platform. Also, most preferably, the device is tamper-resistant.

In accordance with a first aspect, the present invention provides computing apparatus comprising mounted on an assembly main processing means and main memory means, each being connected for communication with one or more other components on the assembly,

characterised by further comprising a trusted device mounted on the assembly and being connected for communications with one or more other components on the assembly, the trusted device being arranged to acquire a true value of an integrity metric of the computing apparatus.

As used herein for reasons of simplicity of description, the term "device" also encompasses plural devices having equivalent function, or equivalent functionality integrated into one or more existing platform devices or assemblies. Additionally, the term 'true' as used herein implies that the value is that which correctly reflects the state of the computing apparatus. This may be ensured if the measurement method is substantially un-modifiable other than by the trusted device.

In accordance with a second aspect, the present invention provides a method of operating a system comprising trusted computing apparatus and a user, the trusted computing apparatus incorporating a trusted device being arranged to acquire the true value of an integrity metric of the computing apparatus, the method comprising the steps of:

the trusted device acquiring the true value of the integrity metric of the trusted computing apparatus;

the user generating a challenge for the trusted computing apparatus to prove its integrity and submitting the challenge to the trusted computing apparatus;

the trusted computing apparatus receiving the challenge, and the trusted device generating a response including the integrity metric and returning the response to the user; and

the user receiving the response, extracting the integrity metric from the response and comparing the integrity metric with an authenticated metric for the trusted computing apparatus that had been generated by a trusted party.

In accordance with a third aspect, the present invention provides a method of establishing a communications channel in a system between trusted computing apparatus and remote computing apparatus, the method including the step of the remote computing apparatus verifying the integrity of the trusted computing apparatus using the above method, and maintaining the communications channel for further transactions in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

In accordance with a fourth embodiment, the present invention provides a method of verifying that trusted computing apparatus is trustworthy for use by a user for processing a particular application, the method including the step of the user verifying the integrity of the trusted computing apparatus using the above method, and the user using the trusted computing apparatus to process the particular application in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

Other aspects and embodiments of the present invention will become apparent from the following description and claims.

Brief Description of the Drawings

A preferred embodiment of the present invention will now be described by way of example only with reference to the accompanying drawings in which:

Figure 1 is a diagram which shows the motherboard of computing apparatus adapted to include a trusted device according to an embodiment of the present invention;

Figure 2 is a diagram which shows in more detail the trusted device shown in Figure 1;

Figure 3 is a diagram which shows in the contents of a certificate stored in the trusted device;

Figure 4 is a diagram, which shows the features of a measurement function responsible for acquiring an integrity metric;

Figure 5 is a flow diagram which illustrates the steps involved in acquiring an integrity metric of the computing apparatus; and

Figure 6 is a flow diagram which illustrates the steps involved in establishing communications between a trusted computing platform and a remote platform including the trusted platform verifying its integrity.

Best Mode For Carrying Out the Invention, & Industrial Applicability

The present exemplary embodiment generally provides the incorporation into a computing platform of a physical trusted device whose function is to bind the identity of the platform to reliably measured data that provides an integrity metric of the platform. The identity and the integrity metric are compared with expected values provided by a trusted party (TP) that is prepared to vouch for the trustworthiness of the platform. If there is a match, the implication is that at least part of the platform is operating correctly, depending on the scope of the integrity metric.

A user verifies the correct operation of the platform before exchanging other data with the platform. A user does this by requesting the trusted device to provide its identity and an integrity metric. (Optionally the trusted device will refuse to provide evidence of identity if it itself was unable to verify correct operation of the platform.) The user receives the proof of identity and the identity metric, and compares them against values which it believes to be true. Those proper values are provided by the TP or another entity that is trusted by the user. If data reported by the trusted device is the same as that provided by the TP, the user trusts the platform. This is because the user trusts the entity. The entity trusts the platform because it has previously validated the identity and determined the proper integrity metric of the platform.

Once a user has established trusted operation of the platform, he exchanges other data with the platform. For a local user, the exchange might be by interacting with some software application running on the platform. For a remote user, the exchange might involve a secure transaction. In either case, the data exchanged is 'signed' by the trusted device. The user can then have greater confidence that data is being exchanged with a platform whose behaviour can be trusted.

The trusted device uses cryptographic processes but does not necessarily provide an external interface to those cryptographic processes. Also, a most desirable implementation would be to make the trusted device tamperproof, to protect secrets by making them inaccessible to other platform functions and provide an environment that is substantially immune to unauthorised modification. Since tamper-proofing is impossible, the best approximation is a trusted device that is tamper-resistant, or tamper-detecting. The trusted device, therefore, preferably consists of one physical component that is tamper-resistant.

Techniques relevant to tamper-resistance are well known to those skilled in the art of security. These techniques include methods for resisting tampering, methods for detecting tampering, and methods for eliminating data when tampering is detected. It will be appreciated that, although tamper-proofing is a most desirable feature of the present invention, it does not enter into the normal operation of the invention and, as such, is beyond the scope of the present invention and will not be described in any detail herein.

The trusted device is preferably a physical one because it must be difficult to forge. It is most preferably tamper-resistant because it must be hard to counterfeit. It typically has an engine capable of using cryptographic processes because it is required to prove identity, both locally and at a distance, and it contains at least one method of measuring some integrity metric of the platform with which it is associated.

Figure 1 illustrates the motherboard 10 of an exemplary computer platform (not shown). The motherboard 10 includes (among other standard components) a main processor 11, main memory 12, a trusted device 14, a data bus 16 and respective standard control lines 17 and address lines 18, and BIOS memory 19 containing the BIOS program for the platform.

Typically, the BIOS program is located in a special reserved memory area, the upper 64K of the first megabyte do the system memory (addresses FØØØh to FFFFh), and the main processor is arranged to look at this memory location first, in accordance with an industry wide standard

The significant difference between the platform and a conventional platform is that, after reset, the main processor is initially controlled by the trusted device, which then hands control over to the platform-specific BIOS program, which in turn initialises all input/output devices as normal. After the BIOS program has executed, control is handed over as normal by the BIOS program to an operating system program, such as Windows NT (TM), which is typically loaded into main memory 12 from a hard disk drive (not shown).

Clearly, this change from the normal procedure requires a modification to the implementation of the industry standard, whereby the main processor 11 is directed to address the trusted device 14 to receive its first instructions. This change may be made simply by hard-coding a different address into the main processor 11. Alternatively, the trusted device 14 may be assigned the standard BIOS program address, in which case there is no need to modify the main processor configuration.

Although, in the preferred embodiment to be described, the trusted device 14 is a single, discrete component, it is envisaged that the functions of the trusted device 14 may alternatively be split into multiple devices on the motherboard, or even integrated into one or more of the existing standard devices of the platform. For example, it is feasible to integrate one or more of the functions of the trusted device into the main processor itself, provided that the functions and their communications cannot be subverted. This, however, would probably require separate leads on the processor for sole use by the trusted functions. Additionally or alternatively, although in the present embodiment the trusted device is a hardware device that is adapted for integration into the motherboard 10, it is anticipated that a trusted device may be implemented as a 'removable' device, such as a dongle, which could be attached to a platform when required. Whether the trusted device is integrated or removable is a matter of design choice.

The trusted device 14 comprises a number of blocks, as illustrated in Figure 2: a controller 20 for controlling the overall operation of the trusted device 14, and interacting with

the other functions on the trusted device 14 and with the other devices on the motherboard 10; a measurement function 21 for acquiring an integrity metric from the platform; a cryptographic function 22 for signing or encrypting specified data; and interface circuitry 23 having appropriate ports (24, 25 & 26) for connecting the trusted device 14 respectively to the data bus 16, control lines 17 and address lines 18 of the motherboard 10. Each of the blocks in the trusted device 14 has access (typically via the controller 20) to appropriate volatile memory areas 27 and/or non-volatile memory areas 28 of the trusted device 14.

For reasons of performance, the trusted device 14 may be implemented as an application specific integrated circuit (ASIC). However, for flexibility, the trusted device is preferably an appropriately programmed micro-controller. Both ASICs and micro-controllers are well known in the art of microelectronics and will not be considered herein in any further detail.

One item of data stored in the non-volatile memory is a certificate 30, which is illustrated in Figure 3. The certificate 30 contains at least a public key 32 of the trusted device 14 and an authenticated value of a platform integrity metric 34 measured by a TP. Optionally, the trusted device 14 also contains an identity (ID) label 36 of the trusted device 14.

Where present, the ID label 36 is a conventional ID label, for example a serial number, that is unique within some context. The ID label 36 is generally used for indexing and labelling of data relevant to the trusted device 14, but is insufficient in itself to prove the identity of the platform under trusted conditions.

The trusted device 14 is equipped with at least one method of reliably measuring some integrity metric of the computing platform with which it is associated. The integrity metric is acquired by the measurement function 21, which is illustrated in more detail in Figure 4.

The measurement function 21 has access to non-volatile memory 40 for storing a hash program 41, plus volatile memory 42 for storing a computed integrity metric 43, in the form of a digest. The hash program 41 contains instructions for computing the digest, in code that is native to the main processor 11. In addition, part of the measurement function 21 is configured to respond to the main processor 11 as if it were addressable memory, such as standard read-only memory, by sensing memory read signals addressed to the trusted device 14 and returning appropriate data. The result is that the main processor 11 sees the trusted device, for the purposes of integrity metric measurement, as a standard read-only memory.

In the preferred implementation, as well as the digest, the integrity metric includes a Boolean value 44, which is stored in volatile memory 45 by the measurement function 21, for reasons that will become apparent.

A preferred process for acquiring an integrity metric will now be described with reference to Figure 5.

In step 500, at switch-on, the measurement function 21 monitors the activity of the main processor 11 on the data, control and address lines (16, 17 & 18) to determine whether the trusted device 14 is the first memory accessed. Under conventional operation, a main processor would first be directed to the BIOS memory first in order to execute the BIOS program. However, in accordance with the present embodiment, the main processor 11 is directed to the trusted device 14, which acts as a memory. In step 505, if the trusted device 14 is the first memory accessed, in step 510, the measurement function 21 writes to non-volatile memory 45 a Boolean value 44, which indicates that the trusted device 14 was the first memory accessed. Otherwise, in step 515, the measurement function writes a Boolean value 44, which indicates that the trusted device 14 was not the first memory accessed.

In the event the trusted device 14 is not the first accessed, there is of course a chance that the trusted device 14 will not be accessed at all. This would be the case, for example, if the main processor 11 were manipulated to run the BIOS program first. Under these circumstances, the platform would operate, but would be unable to verify its integrity on demand, since the integrity metric would not be available. Further, if the trusted device 14 were accessed after the BIOS program had been accessed, the Boolean value 44 would clearly indicate lack of integrity of the platform.

In step 520, when (or if) accessed as a memory by the main processor 11, the main processor 11 reads the stored native hash instructions 41 from the measurement function 21 in step 525. The hash instructions 41 are passed for processing by the main processor 11 over the data bus 16. In step 530, main processor 11 executes the hash instructions 41 and uses them, in step 535, to compute a digest of the BIOS memory 19, by reading the contents of the BIOS memory 19 and processing those contents according to the hash program. In step 540, the main processor 11 writes the computed digest 43 to the appropriate non-volatile memory location 42 in the trusted device 14. The measurement function 21, in step 545, then calls the BIOS program in the BIOS memory 19, and execution continues in a conventional manner.

Clearly, there are a number of different ways in which the integrity metric may be calculated, depending upon the scope of the trust required. The measurement of the BIOS program's integrity provides a fundamental check on the integrity of a platform's underlying

processing environment. Other integrity checks could involve establishing that various other devices, components or apparatus attached to the platform are present and in correct working order. In one example, the BIOS programs associated with a SCSI controller could be verified to ensure communications with peripheral equipment could be trusted. In another example, the integrity of other devices, for example memory devices or co-processors, on the platform could be verified by enacting fixed challenge/response interactions to ensure consistent results. Also, although in the present embodiment the trusted device 14 utilises the data bus as its main means of communication with other parts of the platform, it would be feasible, although not so convenient, to provide alternative communications paths, such as hard-wired paths or optical paths. Further, although in the present embodiment the trusted device 14 instructs the main processor 11 to calculate the integrity metric, it is anticipated that, in other embodiments, the trusted device itself will be arranged to measure one or more integrity metrics.

Preferably, the BIOS boot process includes mechanisms to verify the integrity of the boot process itself. Such mechanisms are already known from, for example, Intel's draft "Wired for Management baseline specification v 2.0 - BOOT Integrity Service", and involve calculating digests of software or firmware before loading that software or firmware. Such a computed digest is compared with a value stored in a certificate provided by a trusted entity, whose public key is known to the BIOS. The software/firmware is then loaded only if the computed value matches the expected value from the certificate, and the certificate has been proven valid by use of the trusted entity's public key. Otherwise, an appropriate exception handling routine is invoked.

Optionally, after receiving the computed BIOS digest, the trusted device 14 may inspect the proper value of the BIOS digest in the certificate and not pass control to the BIOS if the computed digest does not match the proper value. Additionally, or alternatively, the trusted device 14 may inspect the Boolean value 44 and not pass control back to the BIOS if the trusted device 14 was not the first memory accessed.

Figure 6 illustrates the flow of actions by a TP, the trusted device 14 incorporated into a platform, and a user (of a remote platform) who wants to verify the integrity of the trusted platform. It will be appreciated that substantially the same steps as are depicted in Figure 6 are involved when the user is a local user. In either case, the user would typically rely on some form of software application to enact the verification. It would be possible to run the software application on the remote platform or the trusted platform. However, there is a chance that, even on the remote platform, the software application could be subverted in some way. Therefore, it is anticipated that, for a high level of integrity, the software

application would reside on a smart card of the user, who would insert the smart card into an appropriate reader for the purposes of verification.

At the first instance, a TP, which vouches for trusted platforms, will inspect the type of the platform to decide whether to vouch for it or not. This will be a matter of policy. If all is well, in step 600, the TP measures the value of integrity metric of the platform. Then, the TP generates a certificate, in step 605, for the platform. The certificate is generated by the TP by appending the trusted device's public key, and optionally its ID label, to the measured integrity metric, and signing the string with the TP's private key.

The trusted device 14 can subsequently prove its identity by using its private key to process some input data received from the user and produce output data, such that the input/output pair is statistically impossible to produce without knowledge of the private key. Hence, knowledge of the private key forms the basis of identity in this case. Clearly, it would be feasible to use symmetric encryption to form the basis of identity. However, the disadvantage of using symmetric encryption is that the user would need to share his secret with the trusted device. Further, as a result of the need to share the secret with the user, while symmetric encryption would in principle be sufficient to prove identity to the user, it would insufficient to prove identity to a third party, who could not be entirely sure the verification originated from the trusted device or the user.

In step 610, the trusted device 14 is initialised by writing the certificate 30 into the appropriate non-volatile memory locations of the trusted device 14. This is done, preferably, by secure communication with the trusted device 14 after it is installed in the motherboard 10. The method of writing the certificate to the trusted device 14 is analogous to the method used to initialise smart cards by writing private keys thereto. The secure communications is supported by a 'master key', known only to the TP, that is written to the trusted device (or smart card) during manufacture, and used to enable the writing of data to the trusted device 14; writing of data to the trusted device 14 without knowledge of the master key is not possible.

At some later point during operation of the platform, for example when it is switched on or reset, in step 615, the trusted device 14 acquires and stores the integrity metric 42 of the platform.

When a user wishes to communicate with the platform, in step 620, he creates a nonce, such as a random number, and, in step 625, challenges the trusted device 14 (the operating system of the platform, or an appropriate software application, is arranged to recognise the challenge and pass it to the trusted device 14, typically via a BIOS-type call, in an appropriate fashion). The nonce is used to protect the user from deception caused by

replay of old but genuine signatures (called a 'replay attack') by untrustworthy platforms. The process of providing a nonce and verifying the response is an example of the well-known 'challenge/response' process.

In step 630, the trusted device 14 receives the challenge and creates a digest of the measured integrity metric and the nonce, and optionally its ID label. Then, in step 635, the trusted device 14 signs the digest, using its private key, and returns the signed digest, accompanied by the certificate 30, to the user.

In step 640, the user receives the challenge response and verifies the certificate using the well known public key of the TP. The user then, in step 650, extracts the trusted device's 14 public key from the certificate and uses it to decrypt the signed digest from the challenge response. Then, in step 660, the user verifies the nonce inside the challenge response. Next, in step 670, the user compares the computed integrity metric, which it extracts from the challenge response, with the proper platform integrity metric, which it extracts from the certificate. If any of the foregoing verification steps fails, in steps 645, 655, 665 or 675, the whole process ends in step 680 with no further communications taking place.

Assuming all is well, in steps 685 and 690, the user and the trusted platform use other protocols to set up secure communications for other data, where the data from the platform is preferably signed by the trusted device 14.

The techniques of signing, using certificates, and challenge/response, and using them to prove identity, are well known to those skilled in the art of security and will, thus, not be described in any more detail herein.

CLAIMS

1. Computing apparatus comprising mounted on an assembly main processing means and main memory means, each being connected for communication with one or more other components on the assembly,

characterised by further comprising a trusted device mounted on the assembly and being connected for communications with one or more other components on the assembly, the trusted device being arranged to acquire a true value of an integrity metric of the computing apparatus.

2. Computing apparatus according to claim 1, wherein the trusted device comprises device memory means and means for instructing the main processing means to determine the integrity metric and return the integrity metric for storage in the device memory means.

3. Computing apparatus according to claim 2, wherein the means for instructing the main processing means comprises, stored in the device memory means, program code native to the main processing means, and the trusted device is arranged to transfer the instructions of the program code to the main processing means.

4. Computing apparatus according to claim 3, wherein the platform is arranged to cause the instructions to be the first instructions executed after release from reset.

5. Computing apparatus according to claim 3 or claim 4, wherein the trusted device is arranged to transfer the instructions to the main processing means in response to memory read signals from the main processing means.

6. Computing apparatus according to any one of claims 1 to 5, wherein the trusted device comprises device memory means and is arranged to monitor the data bus means and store in the device memory means a flag in the event the first memory read signals generated by the main processing means after the computing apparatus is released from reset are addressed to the trusted device.

7. Computing apparatus according to any one of claims 1 to 6, wherein the trusted device has stored in device memory means at least one of:

a unique identity of the trusted device;

an authenticated integrity metric generated by a trusted party; and

a secret.

8. Computing apparatus according to claim 7, wherein the trusted device has stored in device memory means a secret comprising a private asymmetric encryption key.

9. Computing apparatus according to claim 8, wherein the trusted device also has stored in device memory means a respective public encryption key that has been signed by a trusted party.

10. Computing apparatus according to claim 8 or claim 9, wherein the trusted device has stored in device memory means an authenticated integrity metric generated by a trusted party and includes a encryption function, the trusted device being arranged to generate a response to a received challenge, the response comprising an acquired integrity metric and the authenticated integrity metric, both signed by the encryption function using the private asymmetric encryption key.

11. A trusted device configured for use in computing apparatus according to any one of the preceding claims.

12. A method of operating a system comprising trusted computing apparatus and a user, the trusted computing apparatus incorporating a trusted device being arranged to acquire the true value of an integrity metric of the computing apparatus, the method comprising the steps of:

the trusted device acquiring the true value of the integrity metric of the trusted computing apparatus;

the user generating a challenge for the trusted computing apparatus to prove its integrity and submitting the challenge to the trusted computing apparatus;

the trusted computing apparatus receiving the challenge, and the trusted device generating a response including the integrity metric and returning the response to the user; and

the user receiving the response, extracting the integrity metric from the response and comparing the integrity metric with an authenticated metric for the trusted computing apparatus that had been generated by a trusted party.

13. A method according to claim 12, wherein the challenge includes a nonce, the response includes the integrity metric and the nonce, both digitally signed by the trusted device using a encryption algorithm, and the user verifies the integrity metric and the nonce using a respective encryption algorithm.

14. A method according to claim 13, wherein the trusted device uses a private encryption key to sign the integrity metric and the nonce, and the user uses the respective public encryption key to verify the integrity metric and the nonce.

15. A method according to claim 14, wherein the response includes a certificate held by the trusted device, which certificate has been digitally signed by a trusted party using a private encryption key of the trusted party, the certificate including the public encryption key of the trusted device, and the user verifies the certificate using the public encryption key of the trusted party and uses the public encryption key from the certificate to verify the integrity metric and the nonce.

16. A method of establishing a communications channel in a system between trusted computing apparatus and remote computing apparatus, the method including the step of the remote computing apparatus verifying the integrity of the trusted computing apparatus using the method according to any one of claims 12 to 15, and maintaining the communications channel for further transactions in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

17. A method of verifying that trusted computing apparatus is trustworthy for use by a user for processing a particular application, the method including the step of the user verifying the integrity of the trusted computing apparatus using the method according to any one of claims 12 to 15, and the user using the trusted computing apparatus to process the particular application in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

18. Trusted computing apparatus adapted for use in accordance with the method of any one of claims 12 to 17.

19. Remote computing apparatus arranged for use in accordance with claim 16.

20. A trusted device arranged for use in accordance with any one of claims 12 to 17.

21. Computing apparatus configured to receive a trusted device as claimed in claim 11.

ABSTRACT

Trusted Hardware Device in a Computing Platform

In a computing platform, a trusted hardware device (14) is added to the motherboard (10). The trusted hardware device (14) is configured to acquire an integrity metric, for example a hash of the BIOS memory (19), of the computing platform. The trusted hardware device (14) is tamper-resistant, difficult to forge and inaccessible to other functions of the platform. The hash can be used to convince users that that the operation of the platform (hardware or software) has not been subverted in some way, and is safe to interact with in local or remote applications.

In more detail, the main processing unit (11) of the computing platform is directed to address the trusted hardware device (14), in advance of the BIOS memory, after release from 'reset'. The trusted hardware device (14) is configured to receive memory read signals from the main processing unit (11) and, in response, return instructions, in the native language of the main processing unit (11), that instruct the main processing unit to establish the hash and return the value to be stored by the trusted hardware device (14). Since the hash is calculated in advance of any other system operations, this is a relatively strong method of verifying the integrity of the system. Once the hash has been returned, the final instruction calls the BIOS program and the system boot procedure continues as normal.

Whenever a user wishes to interact with the computing platform, he first requests the integrity metric, which he compares with an authentic integrity metric that was measured by a trusted party. If the metrics are the same, the platform is verified and interactions can continue. Otherwise, interaction halts on the basis that the operation of the platform may have been subverted.

Figure 5.

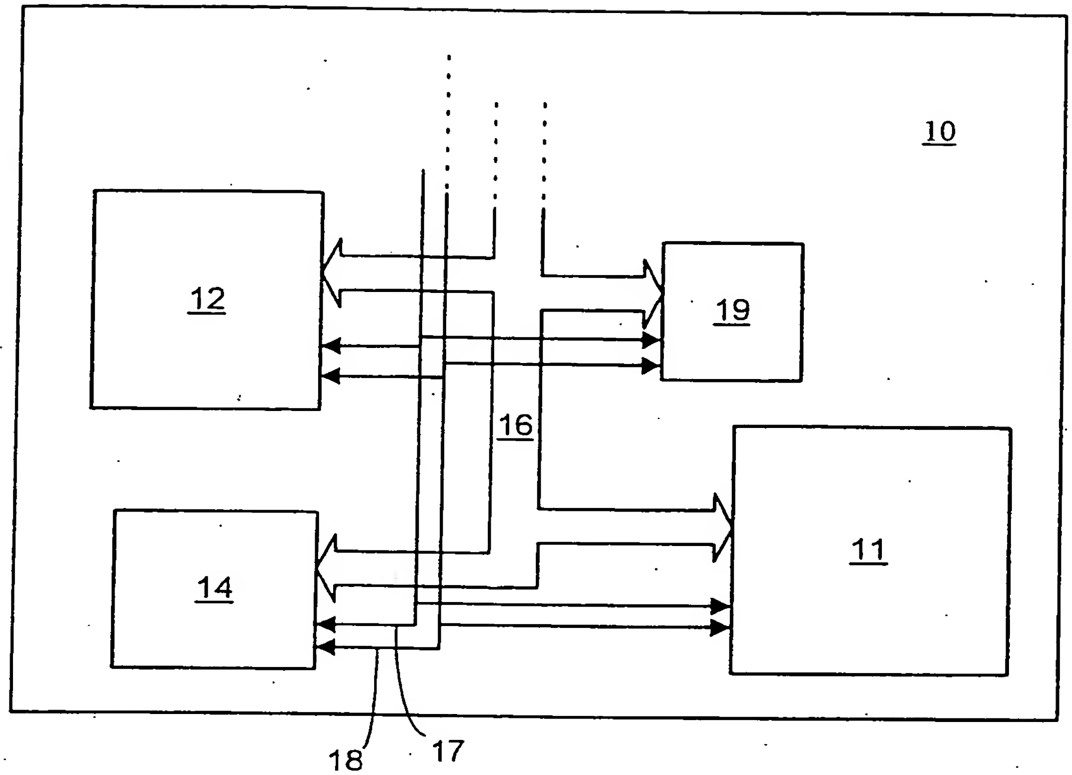


FIGURE 1

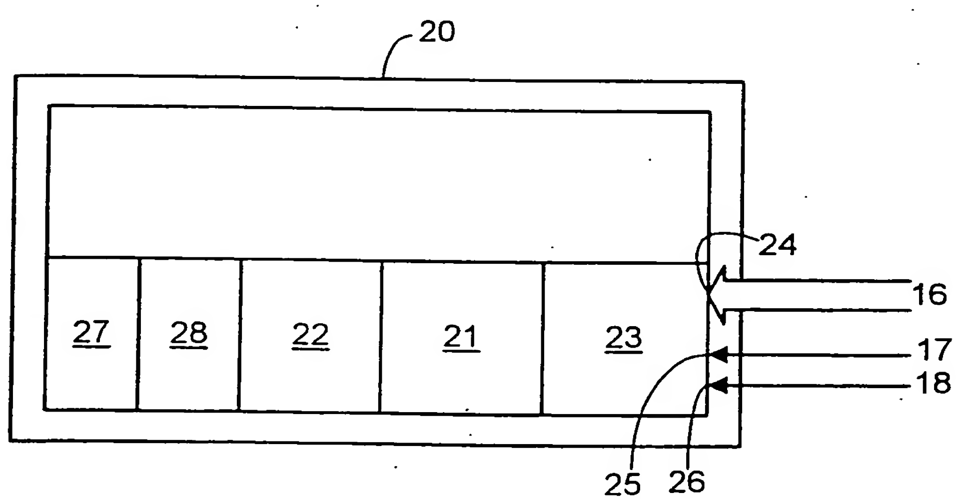


FIGURE 2

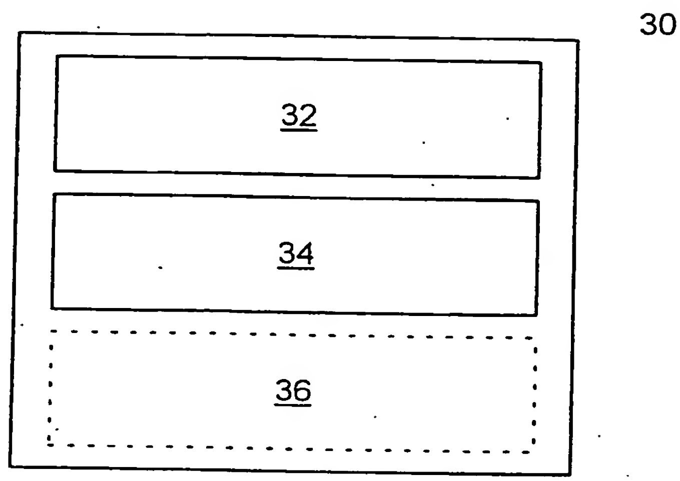


FIGURE 3

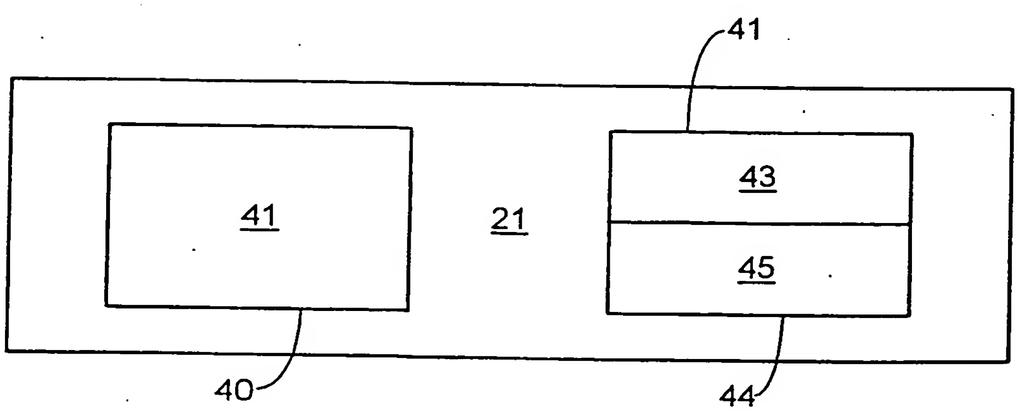


FIGURE 4

This Page Blank (uspto)

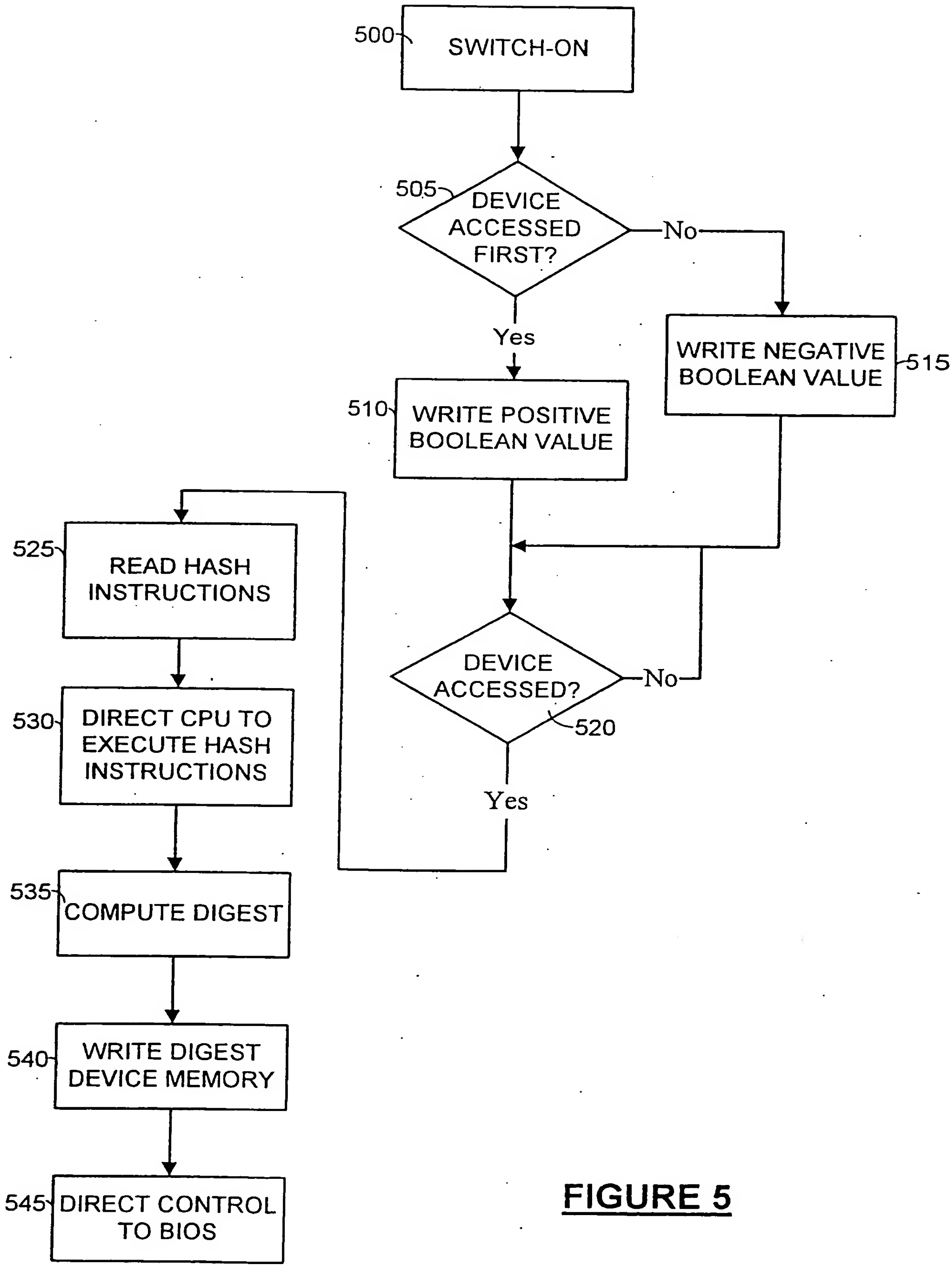

500
SWITCH-ON
505
DEVICE ACCESSED FIRST?
No
Yes
WRITE NEGATIVE BOOLEAN VALUE
515
510
WRITE POSITIVE BOOLEAN VALUE
525
READ HASH INSTRUCTIONS
DEVICE ACCESSED?
No
520
Yes
530
DIRECT CPU TO EXECUTE HASH INSTRUCTIONS
535
COMPUTE DIGEST
540
WRITE DIGEST DEVICE MEMORY
545
DIRECT CONTROL TO BIOS

FIGURE 5

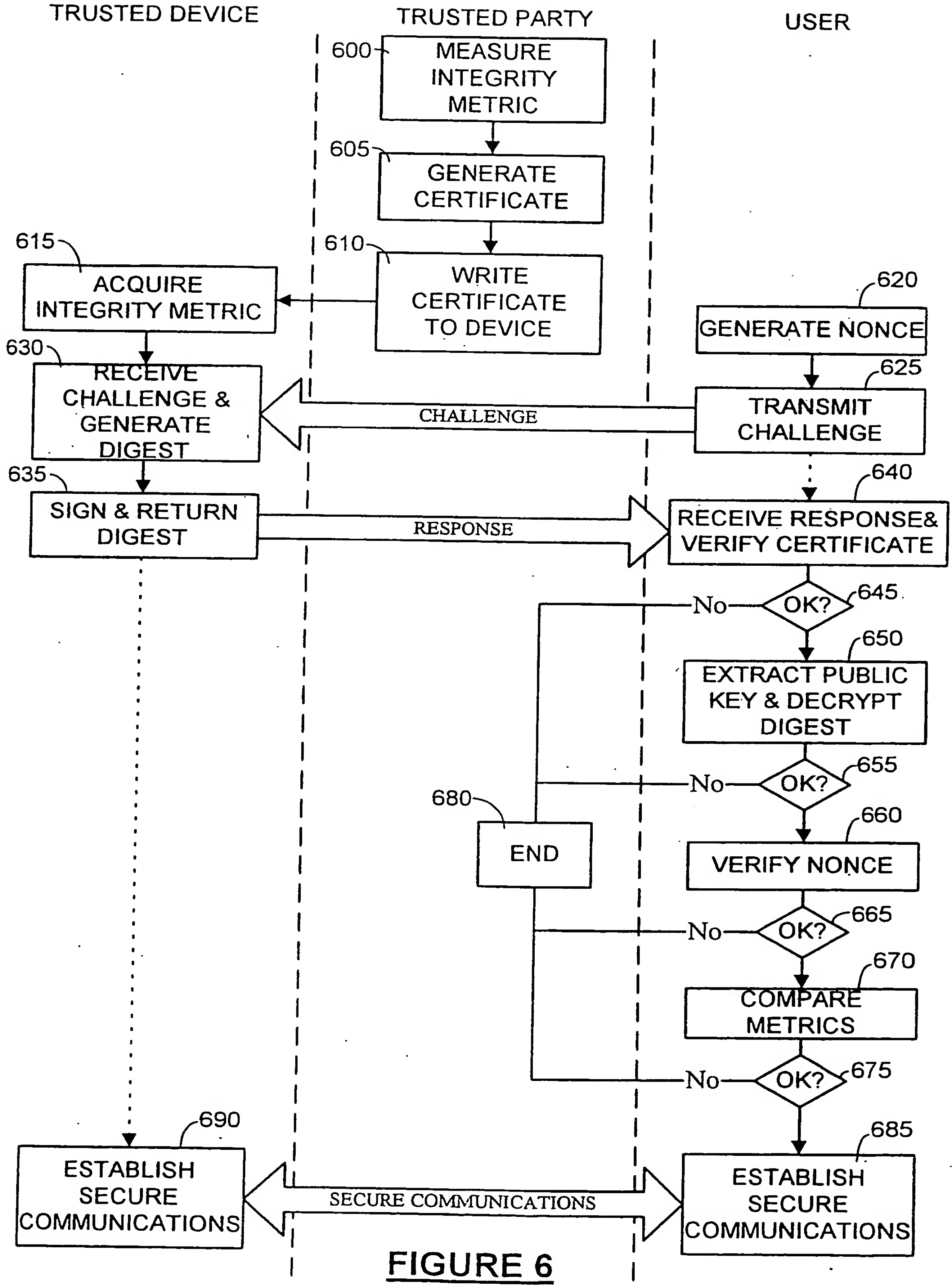

TRUSTED DEVICE
TRUSTED PARTY
USER
600
MEASURE INTEGRITY METRIC
605
GENERATE CERTIFICATE
610
WRITE CERTIFICATE TO DEVICE
615
ACQUIRE INTEGRITY METRIC
620
GENERATE NONCE
625
TRANSMIT CHALLENGE
CHALLENGE
630
RECEIVE CHALLENGE & GENERATE DIGEST
635
SIGN & RETURN DIGEST
RESPONSE
640
RECEIVE RESPONSE& VERIFY CERTIFICATE
645
OK?
No
650
EXTRACT PUBLIC KEY & DECRYPT DIGEST
655
OK?
No
660
VERIFY NONCE
665
OK?
No
670
COMPARE METRICS
675
OK?
No
680
END
685
ESTABLISH SECURE COMMUNICATIONS
690
ESTABLISH SECURE COMMUNICATIONS
SECURE COMMUNICATIONS

FIGURE 6

SYSTEM FOR DIGITALLY SIGNING A DOCUMENT

(HP ref: 30990055)

Technical Field

The present invention relates to apparatus and methods for digitally signing image data, and in particular documents, in a manner which provides a high level of confidence to a signing party that the document they think they are signing is in fact the document they are signing.

Background Art

Conventional prior art mass market computing platforms include the well-known personal computer (PC) and competing products such as the Apple Macintosh™, and a proliferation of known palm-top and laptop personal computers. Generally, markets for such machines fall into two categories, these being domestic or consumer, and corporate. A general requirement for a computing platform for domestic or consumer use is a relatively high processing power, Internet access features, and multi-media features for handling computer games. For this type of computing platform, the Microsoft Windows™ 95 and 98 operating system products and Intel processors, so-called WinTel platforms, dominate the market.

On the other hand, for business use, there are a plethora of available proprietary computer platform solutions available aimed at organizations ranging from small businesses to multi-national organizations. In many of these applications, a server platform provides centralized data storage, and application functionality for a plurality of client stations. For business use, other key criteria are reliability, remote access, networking features, and security features. For such platforms, the Microsoft Windows NT 4.0™ operating system is common, as well as the UNIX and, more recently, the Linux operating systems.

Windows-type operating systems allow a user to run separate applications in separate windows, and provide a so-called WIMP (windows, icons, menus and pointers) interface, whereby a user typically interacts with applications using a keyboard to enter data and a mouse to select options and control applications via dialog boxes and drop-down (or pull-up) menus.

With the increase in commercial activity transacted over the Internet, known as "e-commerce", there has been much interest in the prior art on enabling data transactions between computing platforms, over the Internet. In particular, it is perceived to be important for users to be able to enter into binding contracts over the Internet, without the need for the

current standard hand-signed paper contract. However, because of the potential for fraud and manipulation of electronic data, in such proposals, fully automated transactions with distant unknown parties on a wide-spread scale as required for a fully transparent and efficient market place have so far been held back. The fundamental issue is one of trust between users and their computer platforms, and between interacting computer platforms, for the making of such transactions.

There have been several prior art schemes which are aimed at increasing the security and trustworthiness of computer platforms. Predominantly, these rely upon adding in security features at the application level, that is to say the security features are not inherently embedded in the kernel of operating systems, and are not built in to the fundamental hardware components of the computing platform. Portable computer devices have already appeared on the market which include a smartcard, which contains data specific to a user, which is input into a smartcard reader on the computer. Presently, such smartcards are at the level of being add-on extras to conventional personal computers, and in some cases are integrated into a casing of a known computer. Although these prior art schemes go some way to improving the security of computer platforms, the levels of security and trustworthiness gained by prior art schemes may be considered insufficient to enable widespread application of automated transactions between computer platforms. Before businesses expose significant value transactions to electronic commerce on a widespread scale, they will require greater confidence in the trustworthiness of the underlying technology.

In the applicant's co-pending patent applications 'Trusted Computing Platform' 99301100.6, filed at the European Patent Office on 15 February 1999 and 'Computing Apparatus and Methods of Operating Computing Apparatus' 9905056.9, filed at the UK Patent Office on 5 March 1999, the entire contents of which are incorporated herein by reference, there is disclosed a concept of a 'trusted computing platform' comprising a computing platform which has a 'trusted component' in the form of a built-in hardware component. Two computing entities each provisioned with such a trusted component may interact with each other with a high degree of 'trust'. That is to say, where the first and second computing entities interact with each other the security of the interaction is enhanced compared to the case where no trusted component is present, because:

- A user of a computing entity has higher confidence in the integrity and security of his own computer entity and in the integrity and security of the computer entity belonging to the other party;
- Each entity is confident that the other entity is in fact the entity which it purports to be;

- Where one or both of the entities represent a party to a transaction, e.g. a data transfer transaction, because of the in-built trusted component, third party entities interacting with the entity have a high degree of confidence that the entity does in fact represent such a party;

- The trusted component increases the inherent security of the entity itself, through verification and monitoring processes implemented by the trusted component; and

- The computer entity is more likely to behave in the way it is expected to behave.

As has been indicated above, the conventional method of signing a document is to physically write a signature on the medium (usually paper) upon which an image of a document is reproduced. This method has the advantages that it is clear what is being signed, and the signed image is proof of what was signed. However, it does not meet the needs of e-commerce.

Nowadays it is also possible to digitally sign a document, using a conventional computer platform and standard encryption techniques. In conventional computer platforms, however, the present inventors have appreciated that the electronic rendition of a document which is digitally signed is typically not the same rendition of the document that is visible to the user. It is therefore possible for a user to unintentionally sign data that is different from that which he intended to sign. Conversely, it is also possible for a user to intentionally sign data and later fraudulently claim that the signed data does not correspond to that displayed to him by the computer platform. Such problems would still be the present, even if a trusted platform, as described above, were used.

Conventional electronic methods of signing are well known to those skilled in the art. Essentially, digital data is compressed into a digest, for example by the use of a hash function. Then that digest is encrypted by the use of some encryption method that has been initialised by a secret key (or simply a 'secret'). This is normally done on a computer platform, such as a PC. One implementation is to sign data using a private encryption key held secret on a user's smartcard, which is plugged into a smartcard reader attached to the computer platform. In the specific case of a textual document, the digital data may be the file produced by a word processor application, such as Microsoft's Notepad, Wordpad, or Word. As usual, the act of signing implies that the signer accepts some legal responsibility for the meaning of the data that was signed.

Hash functions are well-known in the prior art and comprise one way functions which are capable of generating a relatively small output data from a relatively large quantity of input data, where a small change in the input data results in a significant change in the output data. Thus, a data file to which is applied a hash function results in a first digest data (the output of the hash function). A small change e.g. a single bit of data in the original data file will result in a significantly different output when the hash function is reapplied to the modified data file. Thus, a data file comprising megabytes of data may be input into the hash function and result in a digital output of the order of 128 to 160 bits length, as the resultant digest data. Having a relatively small amount of digest data generated from a data file stored in the reserved directory is an advantage, since it takes up less memory space and less processing power in the trusted component.

During known signing processes, a user will typically interpret a document as it has been rendered on the computer's monitor at normal magnification and resolution. In existing applications, the user's smartcard signs data in a format that is the representation of the document by the application used to create and/or manipulate the document. The present inventors believe, however, that there is potential for software to send data to the smartcard that has a different meaning from that understood by the user when viewing the screen. This possibility may be sufficient reason to introduce doubt into the validity of conventional methods of digitally signing electronic representations of documents that are to be interpreted by people.

Disclosure of the Invention

The invention consists of systems and methods to improve confidence in digitally signed documents that are to be interpreted by people. They necessarily involve the reliable display of data, which can be used for other purposes.

In accordance with a first aspect, the present invention provides a data processing system arranged to generate a digital signature representative of a document, the data processing system comprising:

main memory means for storing a document to be digitally signed;

main processing means for executing at least one application process comprising means to generate graphics signals for displaying the document;

means for generating a request signal for the document to be signed;

a display system comprising:

frame buffer memory;

means for generating digital image data representative of the document on the basis

of the graphics signals and storing the digital image data in the frame buffer memory; and

means for reading the digital image data from the frame buffer memory, converting the data into signals suitable for displaying an actual image thereof on a display means and forwarding said signals to a display means; and

a trusted component comprising independent processing means operable, in response to receipt of the request signal, for generating a digital signature representative of the digital image data.

As such, the digital signature is generated on the basis of the data used to display the document to a user.

In preferred embodiments, the trusted component comprises means for denying to any unauthorised application or process write access to at least the portion of the frame buffer memory containing the digital image data of the document, and means for generating a digital signature representative of the digital image data while the respective portion of the frame buffer memory is not accessible for writing data to.

In preferred embodiments the data processing system further comprises a removable token comprising processing means for receiving the digital image data, or a representation thereof, from the frame buffer memory and generating a respective digital signature. Conveniently, the removable token is an appropriately programmed smartcard.

In preferred embodiments the trusted component comprises means for acquiring and/or generating trusted image data and means for controlling the display system to highlight the displayed document image using the trusted image data. This provides visual feedback to a user that the trusted component is in control of the operation.

The trusted image data may comprise pixmap data representative of the trusted image or instructions for forming the trusted image.

Preferably, the trusted component comprises means for acquiring and/or generating trusted image data from a removable token.

The trusted component preferably further comprises means for controlling the display system to display messages to a user.

In preferred embodiments, the data processing system further comprises trusted input means by which a user can respond to messages in a secure fashion. The trusted input means may comprise a switch connected to the trusted component via a secure communications channel.

In preferred embodiments, the trusted component and the secure token enact a mutual authentication process in advance of further interactions.

In preferred embodiments, the trusted component forms an integral part of the display

system. For example, the display system may be arranged such that the trusted component is physically and functionally positioned between the main processing means and the frame buffer memory, such that the main processing means can only access the frame buffer memory indirectly through functions of the trusted component.

Preferably, the trusted component further comprises means for generating data summarising a digital signature operation.

Other aspects and embodiments of the invention will become apparent from the following description, claims and drawings.

Brief Description of the Drawings

Embodiments of the present invention will now be described in detail with reference to the accompanying drawings, of which:

Figure 1 is a diagram which illustrates a computer system suitable for operating in accordance with the preferred embodiment of the present invention;

Figure 2 is a diagram which illustrates a hardware architecture of a host computer suitable for operating in accordance with the preferred embodiment of the present invention;

Figure 3 is a diagram which illustrates a hardware architecture of a trusted display processor suitable for operating in accordance with the preferred embodiment of the present invention;

Figure 4 is a diagram which illustrates a hardware architecture of a smart card processing engine suitable for operating in accordance with the preferred embodiment of the present invention;

Figure 5 is a diagram which illustrates a functional architecture of a host computer including a trusted display processor and a smart card suitable for operating in accordance with the preferred embodiment of the present invention;

Figure 6 is a flow diagram which illustrates the steps involved in generating an individual signature of a document;

Figure 7 is diagram which illustrates the sequence of messages between the trusted display processor and the smart card in order to recover seal image data from the smart card;

Figure 8 is diagram which illustrates the sequence of messages between the trusted display processor and the smart card in order to generate a signature of a document image;

Figure 9 is diagram which illustrates the sequence of messages between the trusted display processor and the smart card in order to generate a signature of a summary of the document image signing process;

Figure 10a is a diagram which illustrates an exemplary trusted image;

Figures 10b to 10d are diagrams which illustrate the visual steps in signing a document image; and

Figures 10e to 10g are diagrams which illustrate alternative ways of highlighting the image of a document to be signed.

Best Mode For Carrying Out the Invention, & Industrial Applicability

The preferred embodiment utilises a trusted component that most conveniently uses some of the characteristics of the 'trusted component' described in the applicant's co-pending European patent application number 99301100.6. In that application, the trusted component is a hardware device, comprising a processor programmed to measure an integrity metric of its host computer, compare it with a true value of the integrity metric and communicate the integrity (or otherwise) of the host computer to users or other host computers. The significant similarities between that trusted component and the trusted component in the preferred embodiment herein are:

that they both use cryptographic processes but preferably do not provide an external interface to those cryptographic processes;

that they are both tamper-resistant or tamper-detecting, so that their operation cannot be subverted, at least without the knowledge of the legitimate user; and

that they both preferably consist of one physical hardware component that is both physically and functionally independent of the host computer on which it resides.

Such independence is achieved by the trusted component having its own processing capability and memory.

Techniques relevant to tamper-resistance are well known to those skilled in the art of security, as described in the applicant's co-pending application. These techniques include methods for fabricating components to resist tampering, methods for detecting tampering, and methods for eliminating data when tampering is detected. It will be appreciated that, although tamper-proofing is a most desirable feature of the present invention, it does not enter into the normal operation of the invention and, as such, is beyond the scope of the present description.

In this description, the term 'trusted', when used in relation to a physical or logical component or an operation or process, implies that the behaviour thereof is predictable under substantially any operating condition and highly resistant to interference or subversion by external agents, such as subversive application software, viruses or physical interference.

The term 'host computer' as used herein refers to a data processing apparatus having

at least one data processor, at least one form of data storage and some form of communications capability for interacting with external entities, such as peripheral devices, users and/or other computers locally or via the Internet. The term 'host computer system' in addition to the host computer itself includes standard external devices, such as a keyboard, mouse and VDU, that attach to the host computer.

The term 'document', as used herein, includes any set of data that can be visualised using a host computer system. Commonly a document will be a textual document, such as a contract. However, a document may comprise graphics, or pictures, instead of, or as well as, text. In general, a document may comprise a single page or multiple pages.

The term 'pixmap', as used herein, is used broadly to encompass data defining either monochrome or colour (or greyscale) images. Whereas the term 'bitmap' may be associated with a monochrome image only, for example where a single bit is set to one or zero depending on whether a pixel is 'on' or 'off', 'pixmap' is a more general term, which encompasses both monochrome and colour images, where colour images may require up to 24 bits or more to define the hue, saturation and intensity of a single pixel.

As will become apparent, the trusted component according to the preferred embodiment herein provides a secure user interface and, in particular, controls at least some of the display functionality of its host computer. The trusted component herein may or may not also acquire integrity metrics according to the trusted component in applicant's co-pending patent application, although such acquisition of integrity metrics will not be considered herein.

In essence, the preferred embodiment enables a user to digitally sign a document stored on a host computer using the private key of the user's smartcard, or other form of secure token such as a cryptographic co-processor. The signing is enacted by a trusted display processor (i.e. the trusted component) of the host computer under conditions that provide the user with a high level of confidence that the document being viewed on screen is in fact the document the smartcard is signing. In particular, the smartcard carries trusted image data, or a 'seal', which is passed to the host computer over a secure channel and displayed by the trusted component during the signing procedure. It is in part the display of the trusted image, which is typically unique to the user, which provides the user with the confidence that the trusted component is in control of the signing operation. In addition, in the preferred embodiment, the host computer provides a trusted input device, connected directly to the trusted display processor, by which the user can interact with the host computer in a manner which cannot be subverted by other functions of the host computer.

More particularly, the trusted display processor or a device with similar properties is

associated with video data at a stage in the video processing beyond the point where data can be manipulated by standard host computer software. This allows the trusted display processor to display data on a display surface without interference or subversion by the host computer software. Thus, the trusted display processor can be certain what image is currently being displayed to the user. This is used to unambiguously identify the image (pixmap) that a user is signing. A side-effect of this is that the trusted display processor may reliably display any of its data on the display surface, including, for example, the integrity metrics of the prior patent application, or user status messages or prompts.

Figure 1 illustrates a host computer system according to the preferred embodiment, in which the host computer is a Personal Computer, or PC, which operates under the Windows NT™ operating system. According to Figure 1, the host computer 100 is connected to a visual display unit (VDU) 105, a keyboard 110, a mouse 115 and a smartcard reader 120, and a local area network (LAN) 125, which in turn is connected to the Internet 130. Herein, the smartcard reader is an independent unit, although it may be an integral part of the keyboard. In addition, the host computer has a trusted input device, in this case a trusted switch 135, which is integrated into the keyboard. The VDU, keyboard, mouse, and trusted switch can be thought of as the human/computer interface (HCI) of the host computer. More specifically, the trusted switch and the display, when operating under trusted control, as will be described, can be thought of as a 'trusted user interface'. Figure 1 also illustrates a smartcard 122 for use in the present embodiment as will be described.

Figure 2 shows a hardware architecture of the host computer of Figure 1.

According to Figure 2, the host computer 100 comprises a central processing unit (CPU) 200, or main processor, connected to main memory, which comprises RAM 205 and ROM 210, all of which are mounted on a motherboard 215 of the host computer 100. The CPU in this case is a Pentium™ processor. The CPU is connected via a PCI (Peripheral Component Interconnect) bridge 220 to a PCI bus 225, to which are attached the other main components of the host computer 100. The bus 225 comprises appropriate control, address and data portions, which will not be described in detail herein. For a detailed description of Pentium processors and PCI architectures, which is beyond the scope of the present description, the reader is referred to the book, "The Indispensable PC Hardware Handbook", 3rd Edition, by Hans-Peter Messmer, published by Addison-Wesley, ISBN 0-201-40399-4. Of course, the present embodiment is in no way limited to implementation using Pentium processors, Windows™ operating systems or PCI buses.

The other main components of the host computer 100 attached to the PCI bus 225 include: a SCSI (small computer system interface) adaptor connected via a SCSI bus 235 to

a hard disk drive 240 and a CD-ROM drive 245; a LAN (local area network) adaptor 250 for connecting the host computer 100 to a LAN 125, via which the host computer 100 can communicate with other host computers (not shown), such as file servers, print servers or email servers, and the Internet 130; an IO (input/output) device 225, for attaching the keyboard 110, mouse 115 and smartcard reader 120; and a trusted display processor 260. The trusted display processor handles all standard display functions plus a number of further tasks, which will be described in detail below. 'Standard display functions' are those functions that one would normally expect to find in any standard host computer 100, for example a PC operating under the Windows NT™ operating system, for displaying an image associated with the operating system or application software. It should be noted that the keyboard 110 has a connection to the IO device 255, as well as a direct connection to the trusted display processor 260.

All the main components, in particular the trusted display processor 260, are preferably also integrated onto the motherboard 215 of the host computer 100, although, sometimes, LAN adapters 250 and SCSI adapters 230 can be of the plugin type.

Figure 3 shows a preferred physical architecture for the trusted display processor 260. In accordance with the preferred embodiment, the trusted display processor 260 is a single hardware component having the characteristics of a trusted component, providing the standard display functions of a display processor and the extra, non-standard functions for generating digital signatures and providing a trusted user interface. The skilled person will appreciate that the functions could alternatively be physically split into two or more separate physical components. However, it will be appreciated on reading the following description that integration of all functions into a single trusted component provides a most elegant and convenient solution.

According to Figure 3, the trusted display processor 260 comprises:

a microcontroller 300;

non-volatile memory 305, for example flash memory, containing respective control program instructions (i.e. firmware) for controlling the operation of the microcontroller 300 (alternatively, the trusted display processor 260 could be embodied in an ASIC, which would typically provide greater performance and cost efficiency in mass production, but would generally be more expensive to develop and less flexible);

an interface 310 for connecting the trusted display processor 260 to the PCI bus for receiving image data (i.e. graphics primitives) from the CPU 200 and also trusted image data from the smartcard 122, as will be described;

frame buffer memory 315, which comprises sufficient VRAM (video RAM) in which to

store at least one full image frame (a typical frame buffer memory 315 is 1-2 Mbytes in size, for screen resolutions of 1280x768 supporting up to 16.7 million colours);

a video DAC (digital to analogue converter) 320 for converting pixmap data into analogue signals for driving the (analogue) VDU 105, which connects to the video DAC 320 via a video interface 325;

an interface 330 for receiving signals directly from the trusted switch 135;

volatile memory 335, for example DRAM (dynamic RAM) or more expensive SRAM (static RAM), for storing state information, particularly received cryptographic keys, and for providing a work area for the microcontroller 300;

a cryptographic processor 340, comprising hardware cryptographic accelerators and/or software, arranged to provide the trusted display processor 260 with a cryptographic identity and to provide authenticity, integrity and confidentiality, guard against replay attacks, make digital signatures, and use digital certificates, as will be described in more detail below; and

non-volatile memory 345, for example flash memory, for storing an identifier I_{DP} of the trusted display processor 260 (for example a simple text string name), a private key S_{DP} of the trusted display processor 260, a certificate $Cert_{DP}$ signed and provided by a trusted third party certification agency, such as VeriSign Inc., which binds the trusted display processor 260 with a signature public-private key pair and a confidentiality public-private key pair and includes the corresponding public keys of the trusted display processor 260.

A certificate typically contains such information, but not the public key of the CA. That public key is typically made available using a 'Public Key Infrastructure' (PKI). Operation of a PKI is well known to those skilled in the art of security.

The certificate $Cert_{DP}$ is used to supply the public key of the trusted display processor 260 to third parties in such a way that third parties are confident of the source of the public key and that the public key is a part of a valid public-private key pair. As such, it is unnecessary for a third party to have prior knowledge of, or to need to acquire, the public key of the trusted display processor 260.

The trusted display processor 260 lends its identity and trusted processes to the host computer and the trusted display processor has those properties by virtue of its tamper-resistance, resistance to forgery, and resistance to counterfeiting. Only selected entities with appropriate authentication mechanisms are able to influence the processes running inside the trusted display processor 260. Neither an ordinary user of the host computer, nor any ordinary user or any ordinary entity connected via a network to the host computer may access or interfere with the processes running inside the trusted display processor 260. The

trusted display processor 260 has the property of being "inviolate".

Originally, the trusted display processor 260 is initialised with its identity, private key and certificate by secure communication with the trusted display processor 260 after it is installed onto the motherboard of the host computer 100. The method of writing the certificate to the trusted display processor 260 is analogous to the method used to initialise smartcards by writing private keys thereto. The secure communications is supported by a 'master key', known only to the trusted third party (and to the manufacturer of the host computer 100), that is written to the trusted display processor 260 during manufacture, and used to enable the writing of data to the trusted display processor 260. Thus, writing of data to the trusted display processor 260 without knowledge of the master key is not possible.

It will be apparent from Figure 3 that the frame buffer memory 315 is only accessible by the trusted display processor 260 itself, and not by the CPU 200. This is an important feature of the preferred embodiment, since it is imperative that the CPU 200, or, more importantly, subversive application programs or viruses, cannot modify the pixmap during a trusted operation. Of course, it would be feasible to provide the same level of security even if the CPU 200 could directly access the frame buffer memory 315, as long as the trusted display processor 260 were arranged to have ultimate control over when the CPU 200 could access the frame buffer memory 315. Obviously, this latter scheme would be more difficult to implement.

A typical process by which graphics primitives are generated by a host computer 100 will now be described by way of background. Initially, an application program, which wishes to display a particular image, makes an appropriate call, via a graphical API (application programming interface), to the operating system. An API typically provides a standard interface for an application program to access specific underlying display functions, such as provided by Windows NT™, for the purposes of displaying an image. The API call causes the operating system to make respective graphics driver library routine calls, which result in the generation of graphics primitives specific to a display processor, which in this case is the trusted display processor 260. These graphics primitives are finally passed by the CPU 200 to the trusted display processor 260. Example graphics primitives might be 'draw a line from point x to point y with thickness z' or 'fill an area bounded by points w, x, y and z with a colour a'.

The control program of the microcontroller 300 controls the microcontroller to provide the standard display functions to process the received graphics primitives, specifically:

receiving from the CPU 200 and processing graphics primitives to form pixmap data which is directly representative of an image to be displayed on the VDU 105 screen, where

the pixmap data generally includes intensity values for each of the red, green and blue dots of each addressable pixel on the VDU 105 screen;

storing the pixmap data into the frame buffer memory 315; and

periodically, for example sixty times a second, reading the pixmap data from the frame buffer memory 315, converting the data into analogue signals using the video DAC and transmitting the analogue signals to the VDU 105 to display the required image on the screen.

Apart from the standard display functions, the control program includes a function to mix display image data deceived from the CPU 200 with trusted image data to form a single pixmap. The control program also manages interaction with the cryptographic processor and the trusted switch 135.

The trusted display processor 260 forms a part of the overall 'display system' of the host computer 100; the other parts typically being display functions of the operating system, which can be 'called' by application programs and which access the standard display functions of the graphics processor, and the VDU 105. In other words, the 'display system' of a host computer 100 comprises every piece of hardware or functionality which is concerned with displaying an image.

As already mentioned, the present embodiment relies on interaction between the trusted display processor 260 and the user's smartcard 122. The processing engine of a smartcard suitable for use in accordance with the preferred embodiment is illustrated in Figure 4. The processing engine comprises a processor 400 for enacting standard encryption and decryption functions, to support digital signing of data and verification of signatures received from elsewhere. In the present embodiment, the processor 400 is an 8-bit microcontroller, which has a built-in operating system and is arranged to communicate with the outside world via asynchronous protocols specified through ISO 7816-3, 4, T=0, T=1 and T=14 standards. The smartcard also comprises non-volatile memory 420, for example flash memory, containing an identifier I_{SC} of the smartcard 122, a private key S_{SC}, used for digitally signing data, and a certificate $Cert_{SC}$, provided by a trusted third party certification agency, which binds the smartcard with public-private key pairs and includes the corresponding public keys of the smartcard 122 (the same in nature to the certificate $Cert_{DP}$ of the trusted display processor 260). Further, the smartcard contains 'seal' data SEAL in the non-volatile memory 420, which can be represented graphically by the trusted display processor 260 to indicate to the user that a process is operating securely with the user's smartcard, as will be described in detail below. In the present embodiment, the seal data SEAL is in the form of an image pixmap, which was originally selected by the user as a

unique identifier, for example an image of the user himself, and loaded into the smartcard 122 using well-known techniques. The processor 400 also has access to volatile memory 430, for example RAM, for storing state information (such as received keys) and providing a working area for the processor 400, and an interface 440, for example electrical contacts, for communicating with a smart card reader.

Seal images can consume relatively large amounts of memory if stored as pixmaps. This may be a distinct disadvantage in circumstances where the image needs to be stored on a smartcard 122, where memory capacity is relatively limited. The memory requirement may be reduced by a number of different techniques. For example, the seal image could comprise: a compressed image, which can be decompressed by the trusted display processor 260; a thumb-nail image that forms the primitive element of a repeating mosaic generated by the trusted display processor 260; a naturally compressed image, such as a set of alphanumeric characters, which can be displayed by the trusted display processor 260 as a single large image, or used as a thumb-nail image as above. In any of these alternatives, the seal data itself may be in encrypted form and require the trusted display processor 260 to decrypt the data before it can be displayed. Alternatively, the seal data may be an encrypted index, which identifies one of a number of possible images stored by the host computer 100 or a network server. In this case, the index would be fetched by the trusted display processor 260 across a secure channel and decrypted in order to retrieve and display the correct image. Further, the seal data could comprise instructions (for example PostScript™ instructions) that could be interpreted by an appropriately programmed trusted display processor 260 to generate an image.

Figure 5 shows the logical relationship between the functions of the host computer 100, the trusted display processor 260 and the smartcard 122, in the context of enacting a trusted signing operation. Apart from logical separation into host computer 100, trusted display processor 260 or smartcard 122 functions, the functions are represented independently of the physical architecture, in order to provide a clear representation of the processes which take part in a trusted signing operation. In addition, the 'standard display functions' are partitioned from the trusted functions by a line x-y, where functions to the left of the line are specifically trusted functions. In the diagram, functions are represented in ovals, and the 'permanent' data (including the document image for the duration of the signing process), on which the functions act, are shown in boxes. Dynamic data, such as state data or received cryptographic keys are not illustrated, purely for reasons of clarity. Arrows between ovals and between ovals and boxes represent respective logical communications paths.

In accordance with Figure 5, the host computer 100 includes: an application process 500, for example a wordprocessor process, which requests the signing of a document; document data 505; an operating system process 510; an API 511 process for receiving display calls from the application process 500; a keyboard process 513 for providing input from the keyboard 110 to the application process 500; a mouse process 514 for providing input from the mouse 115 to the application process 500; and a graphics primitives process 515 for generating graphics primitives on the basis of calls received from the application process via the API 511 process. The API process 511, the keyboard process 513, the mouse process 514 and the graphics primitives process 515 are build on top of the operating system process 510 and communicate with the application process via the operating system process 510.

The remaining functions of the host computer 100 are those provided by the trusted display processor 260. These functions are: a control process 520 for co-ordinating all the operations of the trusted display processor 260, and for receiving graphics primitives from the graphics primitives process and signature requests from the application process 500; a summary process 522 for generating a signed summary representative of a document signing procedure in response to a request from the control process 520; a signature request process 523 for acquiring a digital signature of the pixmap from the smartcard 122; a seal process 524 for retrieving seal data 540 from the smartcard 122; a smartcard process 525 for interacting with the smartcard 122 in order to enact challenge/response and data signing tasks required by the summary process 522, the signature request process 523 and the seal process 524; a read pixmap process 526 for reading stored pixmap data 531 and passing it to the signature request process 523 when requested to do so by the signature request process 523; a generate pixmap process 527 for generating the pixmap data 531 on the basis of graphics primitives and seal image data received from the control process 520; a screen refresh process 528 for reading the pixmap data, converting it into analogue signals and transmitting the signals to the VDU 105; and a trusted switch process 529 for monitoring whether the trusted switch 135 has been activated by the user. The smartcard process 525 has access to the trusted display processor's identity data I_{DP}, private key S_{DP} data and certificate $Cert_{DP}$ data 530. In practice, the smart card and the trusted display processor interact with one another via standard operating system calls.

The smartcard 122 has: seal data 540; a display processor process 542 for interacting with the trusted display processor 260 to enact challenge/response and data signing tasks; smartcard identity data I_{SC}, smartcard private key data S_{SC} and smartcard certificate data $Cert_{SC}$ 543.

A preferred process for signing a document using the arrangement shown in Figures 1 to 5 will now be described with reference to the flow diagram in Figure 6.

Initially, in step 600, the user controls the application process 500 to initiate a 'signature request' for digitally signing a document. The application process 500 may be realised as a dedicated software program or may be an addition, for example a macro, to a standard word processing package such as Microsoft's Word. In either case, neither the signature request nor the application process 500 need to be secure. When the user initiates the signature request, he also specifies the document to be signed, if it is not one which is already filling the whole screen. For example, the document may be displayed across a part of the full screen area or in a particular window. Selection of a particular area on screen is a simple task, which may be achieved in several ways (using a WIMP environment), for example by drawing a user-defined box bounding the area or by simply specifying co-ordinates.

Next, in step 602, the application process 500 calls the control process 520 to sign the image that is being displayed (within a defined area or window) on the screen; the control process 520 receives the call. In parallel, although it is not shown, the control process 520 receives any graphics primitives from the graphics primitives process and forwards them onto the generate pixmap process 527. The call from the application process 500 to sign a document includes the co-ordinates (a,b,c,d) of the edges of the document. Note that this sending of co-ordinates generally enables the signing of the entire surface of the screen, a complete window, or of an arbitrary part of the screen. The application process 500 then waits for the control process 520 to return the signature of the image.

In response to the signature request, in step 604, the control process 520 forces the image that is to be signed to be 'static' from the time of the request until the process has been completed. Herein, 'static' means that the document image cannot be modified other than by the trusted display processor 260. This is so that the user can be certain that what he sees is what he is signing at all times during the process. In the present embodiment, the control process 520 achieves a 'static' display by 'holding-off', or not processing, any further graphics primitives. In some situations, the graphics primitives process (or equivalent) may 'buffer' graphics primitives until the control process 520 is ready to receive further graphics primitives. In other situations, graphics primitives for the image to be signed may simply be lost. Where the document image fills the whole screen, making the image static is simply a case not processing any graphics primitives. However, where the image to be signed forms only a subset, for example a window, of the full screen, the control process 520 needs to determine whether received graphics primitives would affect the 'static' area, and reject ones

that would. As such, the pixmap of the static document image in the frame buffer memory 315 remains unchanged by any instructions from the graphics primitives process, or any other process executing on the CPU 200, while the document image is static.

Once the document image has been made static, in step 606, the control process 520 instructs the generate pixmap process 527, including in the call the co-ordinates (a,b,c,d) provided by the application process 500, to modify the pixmap to highlight the document to be signed, as will be described in more detail below with reference to Figure 10c. Then, in step 608, if a smartcard 122 is not already inserted in the smartcard 122 reader 120, as determined by the smart card process 525, the control process 520 instructs the generate pixmap process to display a graphical message asking the user to insert his smartcard 122. This message is accompanied by a ten second countdown timer COUNT. If the countdown timer expires (i.e. reaches zero), as a result of not receiving the smartcard 122, the control process cancels the signing operation in step 614 and returns an exception signal to the application process 500. In response, the application process 500 displays an appropriate user message in step 616. If the smartcard 122 is inserted in time, or is already present, then the process continues.

Next, in step 618, the control process 520 calls the seal process 524, and the seal process 524 calls the smartcard process 525, to recover the seal data 540 from the smartcard 122. Optionally, the control process 520 calls the generate pixmap process 527 to display another message indicating to the user that recovery of the seal data 540 is being attempted. In steps 618 and 620, the smartcard process 525 of the trusted display processor 260 and the display processor process 542 of the smartcard 122 interact using well known, 'challenge/response' techniques to enact mutual authentication and pass the seal data 540 from the smartcard and back to the control process 520. The details of the mutual authentication process and passing of the seal data 540 will now be described with reference to Figure 7.

According to Figure 7, the smartcard process 525 sends a request REQ1 to the smartcard 122 to return the seal data SEAL 540. The display processor process 542 generates a nonce R_1 and sends it in a challenge to the smartcard process 525. The smartcard process 525 generates a nonce R_2 and concatenates it with nonce R_1, signs the concatenation $R_1||R_2$ with its private key to produce a signature $sS_{DP}(R_1||R_2)$, and returns the concatenation $R_1||R_2$, the signature $sS_{DP}(R_1||R_2)$ and the certificate $Cert_{DP}$ back to the display processor process 542 of the smartcard 122. The display processor process 542 extracts the public key of the trusted display processor 260 from the certificate $Cert_{DP}$, and uses this to authenticate the nonce R_1 and the signature $sS_{DP}(R_1||R_2)$ by comparison with the

concatenation $R_1||R_2$, to prove that the seal request came from the expected trusted display processor 260 and that the trusted display processor 260 is online.

The nonces are used to protect the user from deception caused by replay of old but genuine signatures (called a 'replay attack') by untrustworthy processes.

The display processor process 542 of the smartcard 122 then concatenates R_2 with its seal data SEAL 540, signs the concatenation $R_2||SEAL$ using its private key S_{SC} to produce a signature $sS_{SC}(R_2||SEAL)$, encrypts the seal data SEAL 540 using its private key S_{SC} to produce encrypted seal data 540 $sS_{SC}(SEAL)$, and sends nonce R_2, the encrypted seal data $sS_{SC}(SEAL)$, the signature $sS_{SC}(R_2||SEAL)$ and the smartcard's certificate $Cert_{SC}$ to the smartcard process 525 of the trusted display processor 260. The smartcard process 525 extracts the smartcard's public key from the certificate $Cert_{SC}$ and uses this to verify nonce R_2 and the signature $sS_{SC}(R_2||SEAL)$, decrypt the seal data SEAL 540 from the encrypted seal data 540 $sS_{SC}(SEAL)$ and, finally, return the seal data SEAL 540, via the seal process 524, to the control process 520.

Returning to Figure 6, in step 622, when the control process 520 receives the seal data SEAL 540, it forwards the data to the generate pixmap process 527, and instructs the generate pixmap process 527 to generate a seal image and use it to highlight the document to be signed, as will be described below with reference to Figure 10d. Then, in step 624, the control process 520 instructs the generate pixmap process 527 to display a message to the user asking whether they wish to continue with the signing operation. This message is accompanied by a ten second countdown timer COUNT. If the countdown timer expires, in step 626, as a result of not receiving a response from the user, the control process cancels the signing operation, in step 628, and returns an exception signal to the application process 500. In response, the application process 500 displays an appropriate user message in step 629. If, in step 630, the user responds positively by actuating the trusted switch 135 within the ten second time limit, the process continues. The authorisation to continue could alternatively be supplied over an unreliable channel, rather than by using a trusted switch 135, or even using appropriate software routines, providing a reasonable level of authentication is used. Alternatively, it may be decided that the mere presence of an authentic smartcard may be sufficient authorisation for the signing to occur. Such an alternatives are a matter of security policy.

Next, in step 632, the control process 520 instructs the signature request process 523 to request the signing of the document image; the signature request process 523 calls the read pixmap process 526 to request return of a digest of the pixmap data of the document to be signed; and the read pixmap process 526 reads the respective pixmap data, uses a hash

algorithm to generate a digest D_{PIX} of the pixmap data and returns the digest to the signature request process 523. Additionally, the read pixmap process 526 generates 'display format data' FD, which includes information necessary to reconstruct the image from the pixmap data into a text-based document at a later time (FD is not essential, since the document text may not need to be reconstructed), and returns this also to the signature request process 523. For example, the display format data FD may include the number of pixels on the screen surface and their distribution, such as '1024 by 768', and the font type and size used for the text (if the document is text-based) in the document (at least some of this information may instead, or in addition, be contained in a document 'summary', as will be described below). In steps 634 and 636, the signature request process 523 interacts with the display processor process 542 of the smartcard 122 using well-known challenge/response processes to generate an individual signature of the document, as will now be described in detail with reference to the flow diagram in Figure 8.

According to Figure 8, the smartcard process 525 generates a request REQ2 for the smartcard 122 to generate a signature of the digest D_{PIX} and display format data FD. The display processor process 542 of the smartcard 122 responds by generating a nonce R_3 and sending it to the smartcard process 525 with a challenge to return the digest D_{PIX} and the display format data FD. The smartcard process 525 concatenates the digest D_{PIX} with the display format data FD and nonce R_3, and signs the concatenation $D_{PIX}||FD||R_3$ to produce a signature $sS_{DP}(D_{PIX}||FD||R_3)$. The smartcard process 525 then sends the concatenation $D_{PIX}||FD||R_3$ and its respective signature $sS_{DP}(D_{PIX}||FD||R_3)$ to the display processor process 542 of the smartcard 122. The display processor process 542 uses the trusted display processor's public key (which it has already received in the seal data 540 exchange) to verify the trusted display processor's signature $sS_{DP}(D_{PIX}||FD||R_3)$ and nonce R_3, to prove that the digest is the current image digest. The display processor process 542 signs the digest of the pixmap D_{PIX} and the display format data FD, using its private key, to produce two signatures $sS_{SC}(D_{PIX})$ and $sS_{SC}(FD)$ respectively. The display processor process 542 of the smartcard then returns the signed digest $sS_{SC}(D_{PIX})$ and signed display format data $sS_{SC}(FD)$ to the smartcard process 525 of the trusted display processor 260. The smartcard process 525 next verifies the digest D_{PIX} and display format data FD, using the smartcard's public key (which it already has as a result of the seal data 540 exchange), and verifies the smartcard's signature, to prove that the smartcard is still online.

Returning to Figure 6, in step 638, the smartcard process 525 of the trusted display processor 260 concatenates the pixmap PIX, the smartcard's signed versions of the pixmap digest $sS_{SC}(D_{PIX})$ and display format data $sS_{SC}(FD)$ to form an individual signature

$PIX||sS_{SC}(D_{PIX})||sS_{SC}(FD)$ of the image, and returns it, via the signature request process 523, to the control process 520, which returns the individual signature to the application process 500. The application process 500 stores the individual signature, in step 640, and responds with a further call to the control process 520 to 'summarise the signing' operation in step 642. The purpose of a summary is to complete the signature, as will be described with reference to the flow diagram in Figure 9 and also the example summary below:-

```
1    TC-88503-00.01
2    Access time: Thu 06-May-1999,11:18
3    Pages: 2

4    Image01 | 560 x 414 (187,190) [1024 x 768]
5    -----BEGIN SIGNATURE-----
6    PmftitUGoWZh6SLDgqQAvGZZY47Fp8wx5ZqE5HS8bGrSV3RD7LKw0kyXPY6yhGDpVNUc/R2
     +Gr4mm0LqS/twYuPdskyL4uk3no0w3LG2+f+/vzC4cKMPeY/LhbazZScvhK3CJ+apQxyikj
     cY5rTC563klovOPTBI/IyqZPxRnic=
7    -----END SIGNATURE-----

8    Image02 | 670 x 379 (201,228) [1024 x 768]
9    -----BEGIN SIGNATURE-----
10   UV1w5Rgr5F0iAjvUW4GP28NKAA+tOy42uBbP78JeQ5w20MI1afTYkSNtfn9VykYMPIfZLwM
     7ZZV+4fFttuSgOZI4n5iBkSEwtEj0z6ik/np6paq+01GQZhhJCbq8OaX97Gmdg3AoBq4x+D
     hujmqkCJO+Dz6+x8kE24Z8YFXLPOI=
11   -----END SIGNATURE-----

12   Summary signature:
13   -----BEGIN SIGNATURE-----
14   ciZDZL2+41FsFci2cPjWFsfltkyXrfHBUM1kAEyudaZcVxD3XczTN7txSazInM2deJL9qnA
     een2DW1ZGjplEESNkhoZXj0kT5TYNv2ylYFk01SN+JVF09bmc9GdYLo/hSOWyYG/U29Mzqz
     ktaTdTqY/gPh1GajrSJGqRms+we/c=
15   -----END SIGNATURE-----
```

In step 644, the control process 520 calls the summary process 522 to generate a summary message SUM containing the number of images (two in the example summary above) plus the individual signatures of the images (lines 6 and 10 of the example), a label identifying the trusted display device (line 1 in the example), the current time and date (line 2 in the example) and a signed digest of the summary itself (line 14 in the example). For each image, the summary also includes the size of the image in pixels (e.g. 560x414 for image 1), the offset from the origin of the screen in pixels (e.g. 187,190 for image 1) and the display resolution in pixels (e.g. 1024x768 for image 1).

The summary process 522 then generates a digest of the summary D_{SUM} in step 646 and calls the smartcard process 525 of the trusted display processor 260 to interact with the smartcard 122 using challenge/response processes to generate a signature of the summary digest D_{SUM}, as will now be described with reference to Figure 9.

According to Figure 9, the smartcard process 525 generates a request REQ3 for the smartcard 122 to generate a signature of the digest of the summary D_{SUM}. The display

processor process 542 of the smartcard generates a nonce R_4 and sends it in a challenge to return the digest of the summary D_{SUM}. The smartcard process 525 concatenates the digest D_{SUM} with nonce R_4 and signs the concatenation $D_{SUM}||R_4$ to produce a signature $sS_{DP}(D_{SUM}||R_4)$. The smartcard process 525 of the trusted display processor 260 then sends the concatenation $D_{SUM}||R_4$ and respective signature $sS_{DP}(D_{SUM}||R_4)$ to the display processor process 542. The display processor process 542 then verifies the trusted display processor's signature and nonce R_4, using the trusted display processor's public key (which it already has from the seal data 540 exchange), to prove that the summary is the current summary. Next, the display processor process 542 signs the digest of the summary D_{SUM} using its private key and sends the signed digest $sS_{SC}(D_{SUM})$ to the smartcard process 525. The smartcard process 525 of the trusted display processor 260 verifies the digest and verifies the smartcard's signature, using the smartcard's public key, to prove that the smartcard is still online.

Returning to Figure 6, in step 652, the smartcard process 525 returns the summary SUM concatenated with the signed digest of the summary $sS_{SC}(D_{SUM})$ (to form concatenation $SUM||sS_{SC}(D_{SUM})$), via the summary process 522, to the control process 520, and the control process 520 returns the summary $SUM||sS_{SC}(D_{SUM})$ to the application process 500. The application process 500 receives the summary in step 654.

The individual signature and summary may be used by the application process 500, or any other process running on the host computer 100 in various ways outside the scope of this invention, including as proof of contract, for storage or for transmission to other entities.

Finally, in step 656, the control process 520 unfreezes the display, by recommencing receipt and processing of graphic primitives associated with the document image, and thereby in effect returns control of the display back to the application process 500 or other application software. Alternatively, control may not be handed back to the application process 500 until the user actuates the trusted switch 135 again, typically in response to another user message, which, this time, would not have a timeout period. This would give the user more time to review the static document image before returning the host computer to standard, non-trusted operation.

In order to verify a signed document, both the individual signature $PIX||sS_{SC}(D_{PIX})||sS_{SC}(FD)$ and the summary $SUM||sS_{SC}(D_{SUM})$ must be verified. Such verification methods are well known to those skilled in the art of security. For example, the signature on the digest of the pixmap $sS_{SC}(D_{PIX})$ is verified using the public key of the user, which is publicly available and preferably contained within a digital certificate $Cert_{SC}$ supplied by a certification authority. The verified digest is then compared with a value obtained by re-

computing the digest from the pixmap, where the digest is generated using a standard, well-known and defined hash function. If the match is exact, the signature has been verified. Other signatures, including the summary, are checked in the same way.

A preferred method of enabling a person to verify the wording of the signed document is to translate the pixmap back into an image. This requires an application, or indeed a trusted display processor 260, to load the pixmap data PIX into the frame buffer memory 315 of a respective host computer 100. This allows a person to view the document that the signer has signed.

The stages of highlighting a document to be signed will now be described with reference to Figures 10a to 10d.

In the preferred embodiment, the seal data SEAL comprises a pixmap of a trusted image. For example, as shown in Figure 10a, the pixmap of the seal data 540 defines a 'smiley face' 1000. Figure 10b illustrates an image 1005 of an exemplary document Doc1 to be signed, in a window 1010 of the screen (not shown). As a first highlighting step, after the image has been made static but before the seal data has been received, the trusted display processor 260 highlights the document to be signed by superimposing a frame 1020 around the document image 1005, as illustrated in Figure 10c. Also, where a smartcard 122 is not present, a user message 1030 asking the user to insert his smartcard is displayed accompanied by a ten second countdown timer 1035, as also illustrated in Figure 10c. Next, when the smiley face pixmap image is retrieved from the smartcard 122, the trusted display processor 260 embellishes the frame 1040 with multiple instances 1045, or a mosaic, of the smiley face, as shown in Figure 10d. In addition, as shown in Figure 10d, the trusted display processor 260 generates a further user message 1050, accompanied by a ten second countdown timer 1055, asking the user whether they wish to proceed with the signing process. This embellished frame 1040 both indicates to the user that the correct static image area is being acted on and provides the user with a high level of confidence that the trusted display processor 260 is fully in control of the signing process; the presence of the user's own seal image provides confidence to the user that the message has come from the trusted display processor 260 rather than from some other (possibly subversive) software application or hardware device.

Figures 10e and 10g illustrate alternatives to the 'frame' visual effect illustrated in Figures 10c and 10d. In Figure 10e, four single seal images 1060 are positioned at the corners of the static document image using the co-ordinates provided by the application process 500, to define the static image area. In Figure 10f, the static image is defined by modifying the background thereof to show a single seal image. In Figure 10g, the static

image is defined by modifying the background thereof to show a mosaic of seal images. It is expected that the skilled reader will be able to think of other visual effects by which the static image may be highlighted in the light of the present description. In addition, it may be desirable to include further status messages during the signing operation, for example "Retrieving seal data 540 now....", "Generating document signature now....", etc.

It will be appreciated that the trusted display processor 260 needs to be able to display the seal image(s) and the messages in the correct places on screen. Clearly, the seal image and the message images are temporary, to the extent they appear during the signature process and disappear thereafter. There are well-known, standard display techniques for overlaying a first image with a second image, thereby obscuring a part of the first image, then removing the second image and restoring the portion of the first image that had been obscured. Such techniques are used as a matter of course in normal windows environments, for example, where multiple windows may overlap one another. The trusted display processor 260 is arranged to implement one of more of these standard techniques for the purposes of superimposing the seal image(s) and the message images over the standard display.

In some scenarios, it may be that a document is too large to fit all at once onto the VDU 105 screen and still be easily read by a person. Obviously, for the present embodiment to be practical, it is essential that a user can very clearly read the document before signing it. Therefore, the document can be split into multiple screen pages, each of which needs to be signed and cryptographically chained to the signature of the previous page, as will now be described.

First, the application process 500 causes the image of the first page to be displayed and makes a call to the trusted display processor 260 for signing as before. When the trusted display processor 260 returns the individual signature, instead of requesting a summary, the application process 500 instructs the trusted display processor 260 to display the image of the second page and sign the image. Clearly, in this case, the trusted display processor 260 is arranged to support such a request by the application process 500. Only after all images have been signed and returned to the application process 500 does the application process 500 issue a request for a summary. Then, the summary includes the number of images that were signed in this multi-page document, for example as illustrated in the two-page summary above.

The first page in the multi-page document is signed in the same way as a single page, resulting in return of an individual signature. When subsequent images are presented for signing, however, the trusted display processor 260 recognises that they are part of a multi-

page document because no summary request was received after the previous signature request. As a result, the trusted display processor 260 displays a different message, which requests permission from the user to sign a continuation page. In response, the user who is signing a multi-page document uses the same reliable permission channel as before (for example, the trusted switch 135) to confirm to the trusted display processor 260 that this page is associated with the previous page, and is also to be signed. When the trusted display processor 260 receives this multi-page confirmation, it concatenates the signature of the previous signed page with the pixmap of the current page, creates a digest of the concatenation, and sends that to the smartcard for signing. This is instead of sending a digest of just the current pixmap. This process cryptographically 'chains' a subsequent page to the previous page, so that pages cannot be rearranged without detection, nor can intermediate pages be inserted or deleted without detection.

The validity of the first page may be checked in exactly the same way as a single page. The validity of subsequent pages is checked using the same method as for a single page, except that the digest of the current pixmap is replaced by the digest of the concatenated previous signature and current pixmap.

It will be appreciated that there are many ways of cryptographically chaining a subsequent page to a previous page. Such ways will be obvious to those skilled in the art of security in the light of the present description.

For added security, the image of each page of a multi-page document may be arranged to include the conventional footer 'Page x of y, where 'x' is the number of the page and 'y' is the total number of pages. This enables ready detection by a person of a truncated document simply by reading the document.

A significant benefit of the present document signing scheme is that a signed document can be re-signed and countersigned. As such, it is preferable for the summary of a document to include an audit trail. There are many variations on re-signing and countersigning, although (obviously) an electronic integrity check should always be done before any further signing. At one extreme, the new signer could view, confirm and re-sign each signed image in turn, effectively replacing the original signature by a new one. This method could be used, for example, by a user signing a document prepared for him by someone else. At the other extreme, the new signer could simply 'rubber stamp' the original signature by signing the original summary, without necessarily viewing the document at all. This could be useful to a manager countersigning the work of a trusted employee.

For a re-signing operation, the application process 500 issues a re-signing request, and transmits an already signed document (plus the individual signature(s) and the summary)

to the trusted display processor 260. The trusted display processor 260 verifies the signed document using the public key of the signer, recovers the pixmap of the document (or each page of the document) and displays each verified image in the correct order to the new user, as if they were original images from the signature request application. The user confirms the acceptance of each individual image, for example using a trusted switch 135 as before, and causes the images to be signed as before by a smartcard belonging to the new user. This results in a signed document that is the same as the original document, except that it has been signed by the smartcard belonging to the new user.

Similarly, for a counter-signing operation, the application process 500 issues a counter-signing request and transmits the signed document (plus individual signatures and the summary) to the trusted display processor 260. The trusted display processor 260 verifies the signed document and displays each verified image in the correct order to the new user, as if they were original images from the application process 500. The user confirms acceptance of each individual image and the trusted display processor 260 signs the original summary using the smartcard belonging to the new user. Optionally the new user could provide a certificate of the previous user's public key, signed by the new user, to ease the processing overhead associated with later verification of the signature.

Clearly, there are many possible variations on the theme of re-signing and countersigning, which will be apparent to the skilled person in the light of the present description.

Since a document may have a history of signing, re-signing and/or counter-signing, the present embodiment conveniently provides audit information, which forms part of the document summary. This audit information allows the signature history of the document to be traced. The audit information includes data about the previous state of the document and the actions taken by the new user to create the new state of the document. The audit information is signed by the trusted display processor 260, since the audit information must be independent of the user. The audit information always contains any previous summary information (including the signature on that summary information, by the previous signer). If the signed document has been created from scratch, the identity label I_{DP} of the trusted display processor 260 is inserted as an audit root. The audit information preferably also includes an indication of which individual images were viewed and confirmed by the new user, and whether the document was created from scratch, or was re-signed, or was countersigned by the new user. To create a summary including audit information, the smartcard is sent a digest of the audit information concatenated with the previously described contents of a summary, rather than a digest of just the previously described contents of a

summary. The rest of the process is as previously described.

An enhancement to the process for signing a document is that, prior to signing the pixmap data, the trusted display processor 260 compresses the pixmap using a lossless compression algorithm so that the overheads associated with storing and sending the individual signature are reduced.

The pixmap may be compressed by standard compression algorithms, for example a codeword-based algorithm applying LZ-1 or LZ-2 compression. Alternatively, a technique similar to OCR (optical character recognition) may be used to compress the pixmap. In this case, the situation differs from conventional OCR in that the input data has been perfectly 'scanned', albeit at a lower resolution than in conventional OCR. The OCR-compressed version of the pixmap may be generated by 'blob-matching' to create an alphabet for the pixmap, constructing a pixmap of each character in the alphabet, and constructing a message using those characters, such that the message represents the original pixmap. This means that the pixmap can been compressed to a new alphabet and a message written in that alphabet.. Since there are, obviously, no errors nor ambiguity in the pixmap data, this is a lossless compression method.

Another way of reducing the size of the image pixmap is by representing the image as a pure black and white image, requiring only a single bit – set to zero or one - to define whether a pixel is black or white. Otherwise, the document image is represented as a full colour image, where each pixel may typically require up to 24-bits. Obviously, this technique may be suitable for simple, black and white text-based documents. However, it would not be appropriate for colour documents or images.

At any time, the document image may be converted back into a text-based document using an OCR-type process to reconstruct a standard digital textual representation of the document. This technique cannot be used in the signature, since the textual mapping may be incorrect, but can be used by the receiver of a signed document to convert it back into a standard digital textual representation (such as ASCII) for subsequent machine manipulation. In preferred embodiments, the trusted display processor 260 is equipped to enact OCR document recovery.

To enact OCR, an OCR alphabet is generated in a standard fashion and is then matched to stored fonts and hence converted to a standard character set. As in conventional OCR, ambiguous matches may be retained as a pixmap and flagged for conversion by the user. (This is unlikely, particularly if font type and size information has been supplied in the display format data FD, because there is no error in the data.) In cases of extreme caution, the entire reconstructed document should be manually checked by a person against the view

of the document that the signer intended to sign.

Preferably all document reconstruction processes are done by processes that are trusted.

The preferred embodiment described above relies on the premise that the trusted display processor 260 has direct and exclusive access to video data stored in the frame buffer memory 315, beyond the point where the video data can be manipulated by host computer 100 software, including the operating system. This implies that the video data cannot be modified unless the trusted display processor 260 makes the modification.

It will be appreciated that not all computer architectures are arranged in this way. For example, some computer architectures are arranged such that the frame buffer memory forms a part of the main memory, thus forming a single address space (SAS) display system. One benefit of such a system is that both the CPU and the display processor can access the frame buffer memory and share the graphics operation overhead, thereby improving graphics performance. Clearly, an implementation of the present invention in such a SAS system cannot rely on the premise that the buffer memory is safe during signing, since the CPU can still access the memory. However, there are many ways in which such a SAS system may be modified to support implementations of the present invention. For example, the memory could be provided with a control line from a trusted display processor such that, during a signing operation, the memory is prevented from being updated by data from the CPU. The memory devices themselves are preferably modified so that they include the extra logic to perform this function. Alternatively, access to memory is blocked by other logic circuits inserted into the normal control path of the memory. Such systems, therefore, rely on the modified premise that the video data in the frame buffer memory can only be modified, other than by the trusted display processor, with the permission of the trusted display processor. Clearly, this premise is as valid for secure operation as the first premise, as long as the system is truly secure.

In other architectures, for example in simple graphics environments, the functionality of a display processor may form part of the operating system itself, thereby removing the requirement for separate display processor hardware. Clearly, in this case, the graphics overhead put on the CPU will be higher than in a system with separate display processor hardware, thereby limiting the graphics performance of the platform. Clearly, there is then no place for a 'trusted display processor' as such. However, it will be apparent to the skilled person that the same function as provided by the trusted display processor, that of protecting the frame buffer memory and interacting with a smartcard, can be implemented using an appropriate trusted component, which controls the display system (in whatever form) during

signing.

In other embodiments of the invention, in addition or alternatively, the trusted display processor (or equivalent) includes an interface for driving a trusted display. The trusted display might be, for example, an LCD panel display. In the same way that the trusted switch provides a trusted means for a user to interact with the trusted display processor, the trusted display can provide a trusted means for feeding back information to the user other than via the standard VDU. For example, the trusted display might be used to provide user status messages, as described above, relating to a signing operation. As such, applications running on the standard host computer should not be able to access the trusted display, because the display is connected either directly to the trusted display processor or via some form of trusted channel. In essence, such a trusted display is an addition to the so-called 'trusted interface' described above. In practice, there is no reason why other forms of trusted feedback device, of which the trusted display is one example, could not be included in addition, or as an alternative. For example, there may be scenarios where some form of trusted sound device would be useful for providing audible feedback.

CLAIMS

1. A data processing system arranged to generate a digital signature representative of a document, the data processing system comprising;

main memory means for storing a document to be digitally signed;

main processing means for executing at least one application process comprising means to generate graphics signals for displaying the document;

means for generating a request signal for the document to be signed;

a display system comprising:

frame buffer memory;

means for generating digital image data representative of the document on the basis of the graphics signals and storing the digital image data in the frame buffer memory; and

means for reading the digital image data from the frame buffer memory, converting the data into signals suitable for displaying an actual image thereof on a display means and forwarding said signals to a display means; and

a trusted component comprising independent processing means operable, in response to receipt of the request signal, for generating a digital signature representative of the digital image data.

2. A data processing system according to claim 1, wherein the trusted component comprises means for denying to any unauthorised application or process write access to at least the portion of the frame buffer memory containing the digital image data of the document, and means for generating a digital signature representative of the digital image data while the respective portion of the frame buffer memory is not accessible for writing data to.

3. A data processing system according to claim 1 or claim 2, further comprising a removable token comprising processing means for receiving the digital image data, or a representation thereof, and generating a respective digital signature.

4. A data processing system according to any one of the preceding claims, wherein the trusted component further comprises means for acquiring and/or generating trusted image data and means for controlling the display system to highlight the displayed document image using the trusted image data.

5. A data processing system according to claim 4, wherein the trusted image data comprises pixmap data representative of the trusted image or instructions for forming the trusted image.

6. A data processing system according to claim 4 or claim 5, wherein the trusted component controls the display system to highlight the displayed document image by producing one or more of the following visual effects:

a border, or an indicator or indicators defining a border, characterised by the trusted image and positioned at least partly around the document image;

a background pattern characterised by the trusted image forming at least part of the background of the document image;

an image characterised by the trusted image formed within the document image; and/or

a text message characterised by the trusted image formed within or near the document image.

7. A data processing system according to any one of claims 4 to 6, wherein the trusted component comprises means for acquiring and/or generating trusted image data from a removable token.

8. A data processing system according to any one of the preceding claims, wherein the trusted component further comprises means for controlling the display system to display messages to a user.

9. A data processing system according to claim 8, further comprising trusted input means by which a user can respond to messages in a secure fashion.

10. A data processing system according to claim 9, wherein the trusted input means comprises a switch connected to the trusted component via a secure communications channel.

11. A data processing system according to claim 3 or claim 7, wherein the trusted component and the secure token enact a mutual authentication process in advance of further interactions.

12. A data processing system according to any one of the preceding claims, wherein the

trusted component forms an integral part of the display system.

13. A data processing system according to claim 12, wherein the display system is arranged such that the trusted component is physically and functionally positioned between the main processing means and the frame buffer memory, such that the main processing means can only access the frame buffer memory indirectly through functions of the trusted component.

14. A data processing system according to any one of the preceding claims, further comprising means for generating data summarising a digital signature operation.

15. A method for digitally signing a document, comprising the steps:

generating digital image data of the document and updating the digital image data in a frame buffer memory;

reading the digital image data from the frame buffer memory, converting the digital image data into signals suitable for driving a visual display means and transmitting the signals to a visual display means for displaying an image of the document; and

on demand, reading the digital image data from the frame buffer memory and generating a digital signature representative thereof.

16. A method according to claim 15, further comprising the step of temporarily denying write access to the frame buffer memory by unauthorised processes while generating the digital signature.

17. A method according to claim 14 or claim 15, further comprising the step of acquiring and/or generating trusted image data and using the trusted image data to highlight the document image.

18. A method according to claim 17, wherein step of highlighting the document image is achieved by generating any one or more of the following visual effects:

a border, or an indicator or indicators defining a border, characterised by the trusted image and positioned at least partly around the document image;

a background pattern characterised by the trusted image forming at least part of the background of the document image;

an image characterised by the trusted image formed within the document image; and/or

a text message characterised by the trusted image formed within or near the document image.

19. A method according to claim 17 or claim 18, wherein the trusted image data is acquired from a removable token.

20. A data processing system arranged to digitally sign a document in accordance with any one of claims 15 to 19.

21. A method for digitally signing a document comprising a plurality of individual viewable pages, comprising the steps:

a) generating digital image data of the first page of the document and updating the digital image data in a frame buffer memory;

b) reading the digital image data from the frame buffer memory, converting the digital image data into signals suitable for driving a visual display means and transmitting the signals to a visual display means for displaying an image of the document;

c) reading the digital image data from the frame buffer memory, generating a digital signature representative thereof and storing the digital signature;

iv) repeating steps a) to c) for the other page(s) of the document; and

v) generating a further digital signature representative of all previous digital signatures.

22. A method for a second user to digitally counter-sign a document that has already been signed by a first user, the document being accompanied by a respective first digital signature generated by using a secret of the first user, comprising the steps:

generating digital image data of the document and updating the digital image data in a frame buffer memory;

reading the digital image data from the frame buffer memory, converting the digital image data into signals suitable for driving a visual display means and transmitting the signals to a visual display means for displaying an image of the document;

verifying the integrity of the first digital signature; and

on the basis of a secret of the second user, generating a digital signature representative of the first digital signature.

23. A method for a second user to digitally re-signing a document that has already been

signed by a first user, the document being accompanied by a respective first digital signature generated by using a secret of the first user, comprising the steps:

generating digital image data of the document and updating the digital image data in a frame buffer memory;

reading the digital image data from the frame buffer memory, converting the digital image data into signals suitable for driving a visual display means and transmitting the signals to a visual display means for displaying an image of the document;

verifying the integrity of the first digital signature; and

reading the digital image data from the frame buffer memory and, on the basis of a secret of the second user, generating a digital signature representative of the first digital signature.

24. A data processing system arranged to generate a digital signature representative of a document, the data processing system comprising;

main processing means for generating graphics signals for displaying a document;

a display system comprising:

frame buffer memory;

means for generating digital image data representative of the document on the basis of the graphics signals and storing the digital image data in the frame buffer memory; and

means for reading the digital image data from the frame buffer memory, converting the data into signals suitable for displaying an actual image thereof on a display means and forwarding said signals to a display means; and

means for generating a digital signature representative of the digital image data.

25. A system for digitally signing a document comprising:

frame buffer memory;

means for generating image data representative of a document and storing the image data in the frame buffer memory;

means for reading the image data from the frame buffer memory and displaying a respective image on a display means; and

means for reading the image data from the frame buffer memory and generating a digital signature thereof.

26. A trusted component for use in a data processing system according to any one of claims

1 to 14.

27. A trusted component according to claim 26, fabricated to be tamper resistant.

ABSTRACT

Trusted System For Digitally Signing a Document

The preferred embodiment of the invention comprises a computer system which employs a trusted display processor (260), which has a trusted processor (300) and trusted memory (305, 315, 335, 345) physically and functionally distinct from the processor and memory of the computer system. The trusted display processor (260) is immune to unauthorised modification or inspection of internal data. It is physical to prevent forgery, tamper-resistant to prevent counterfeiting, and has crypto functions (340) to securely communicate at a distance. The trusted display processor (260) interacts with a user's smartcard (122) in order to extract and display a trusted image, or seal (1000), generate a digital signature of the bitmap of a document image and control the video memory (315) so that other processes of the computer system cannot subvert the image during the signing process. The user interacts with the trusted display processor via a trusted switch (135).

Figure 3.

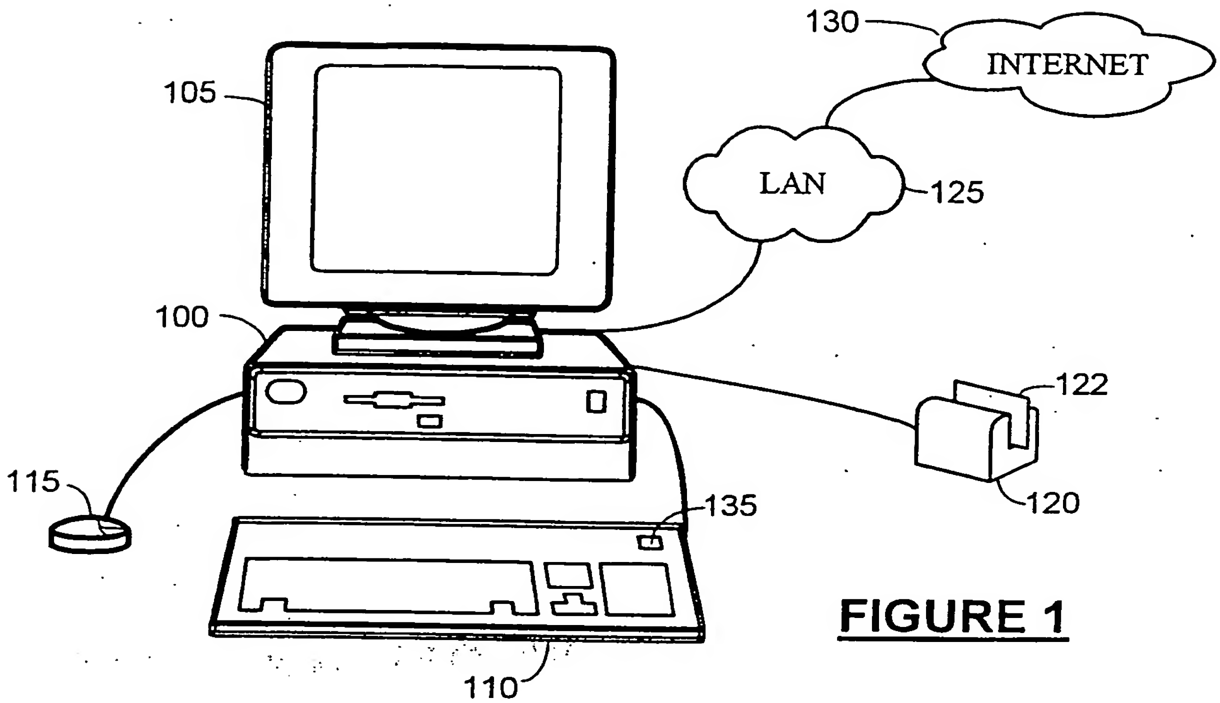


FIGURE 1

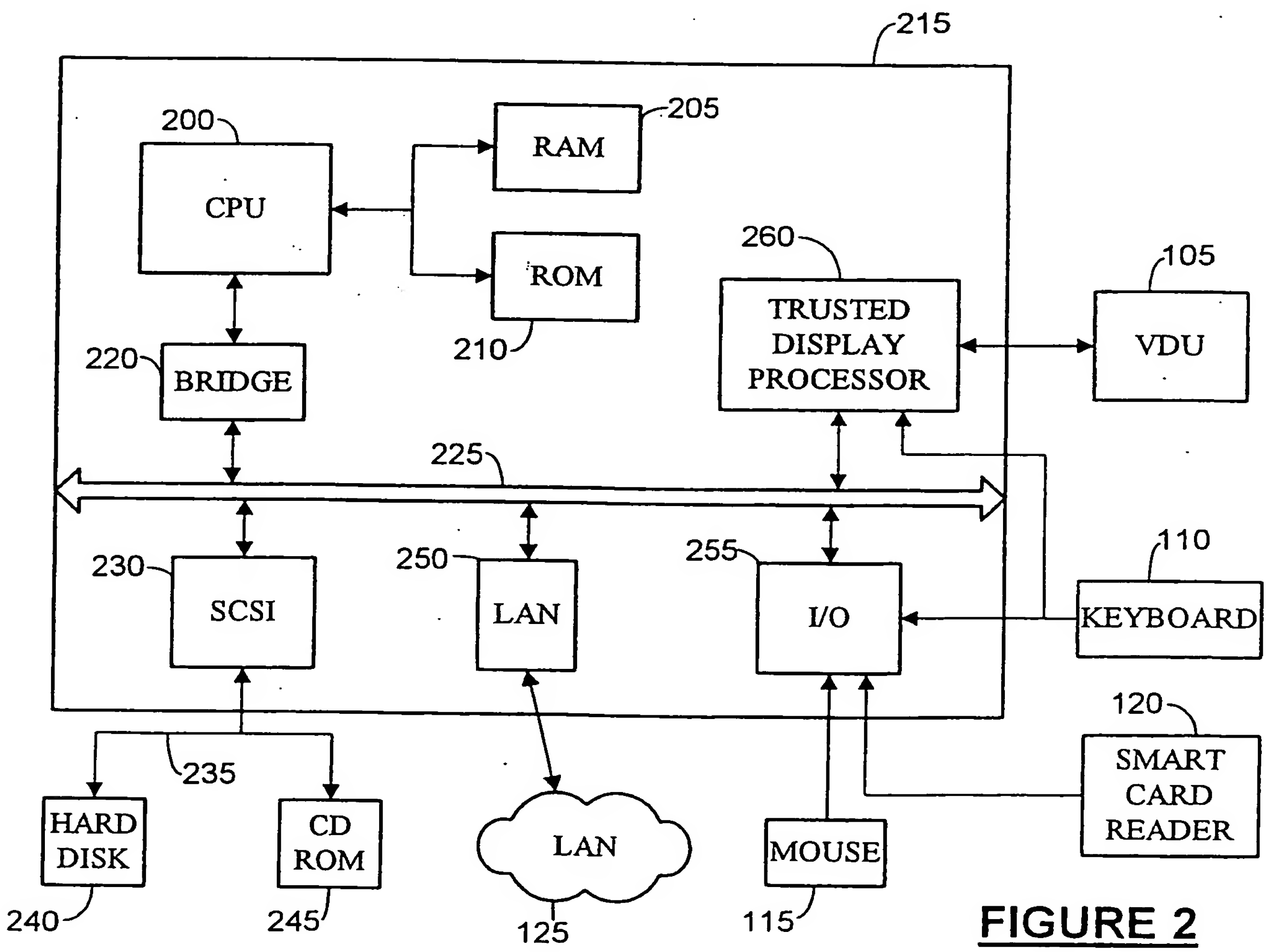


FIGURE 2

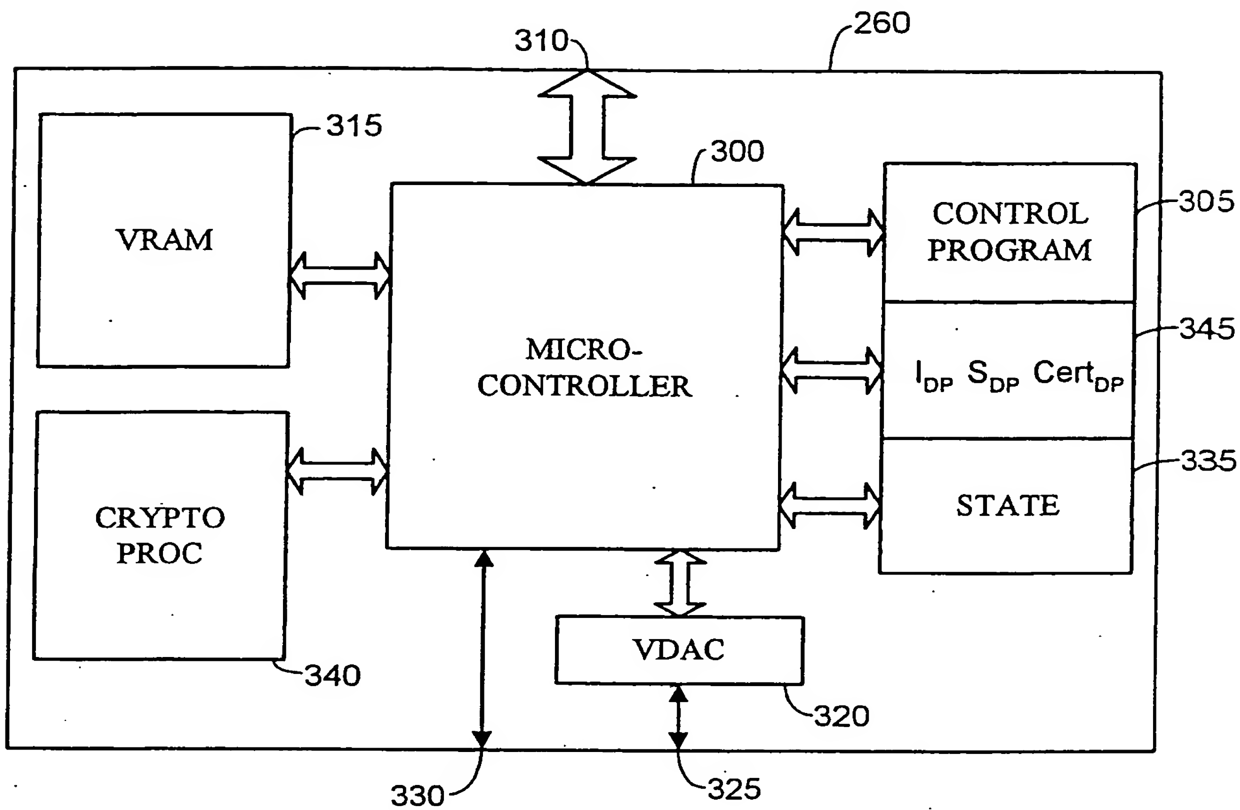


FIGURE 3

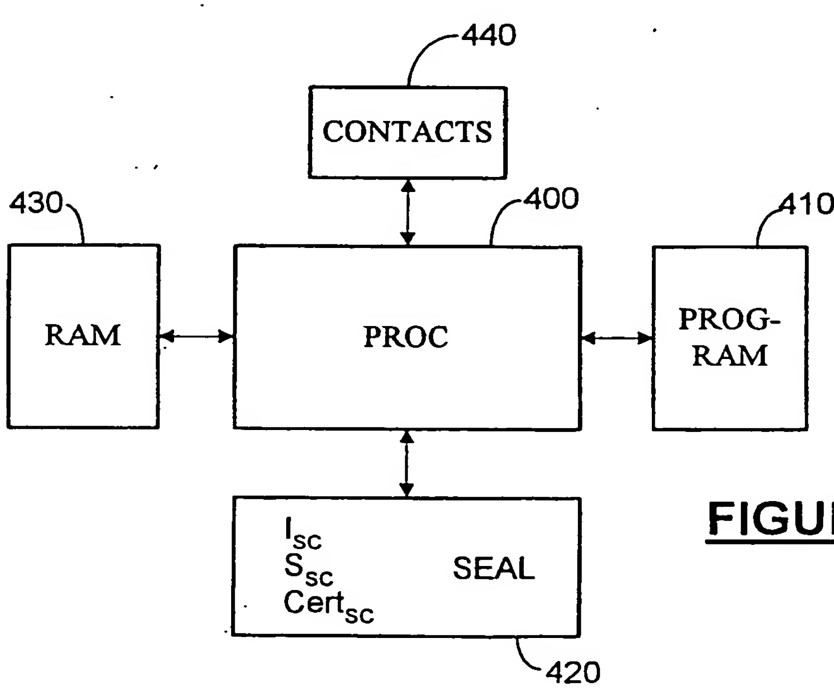


FIGURE 4

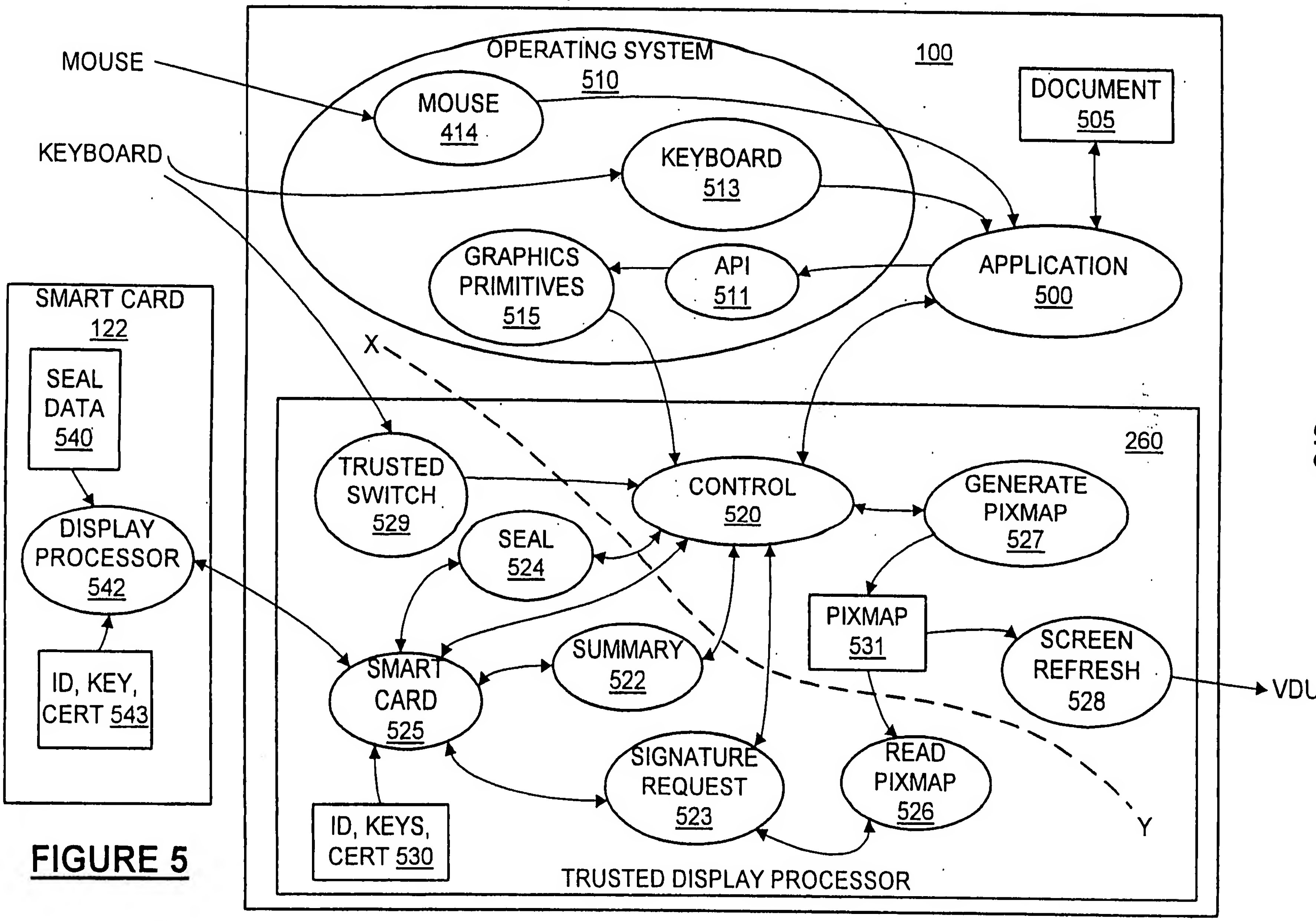

MOUSE
KEYBOARD
OPERATING SYSTEM 510
100
MOUSE 414
KEYBOARD 513
DOCUMENT 505
GRAPHICS PRIMITIVES 515
API 511
APPLICATION 500
SMART CARD 122
SEAL DATA 540
DISPLAY PROCESSOR 542
ID, KEY, CERT 543
X
260
TRUSTED SWITCH 529
CONTROL 520
GENERATE PIXMAP 527
SEAL 524
PIXMAP 531
SCREEN REFRESH 528
VDU
SMART CARD 525
SUMMARY 522
SIGNATURE REQUEST 523
READ PIXMAP 526
Y
ID, KEYS, CERT 530
TRUSTED DISPLAY PROCESSOR

FIGURE 5

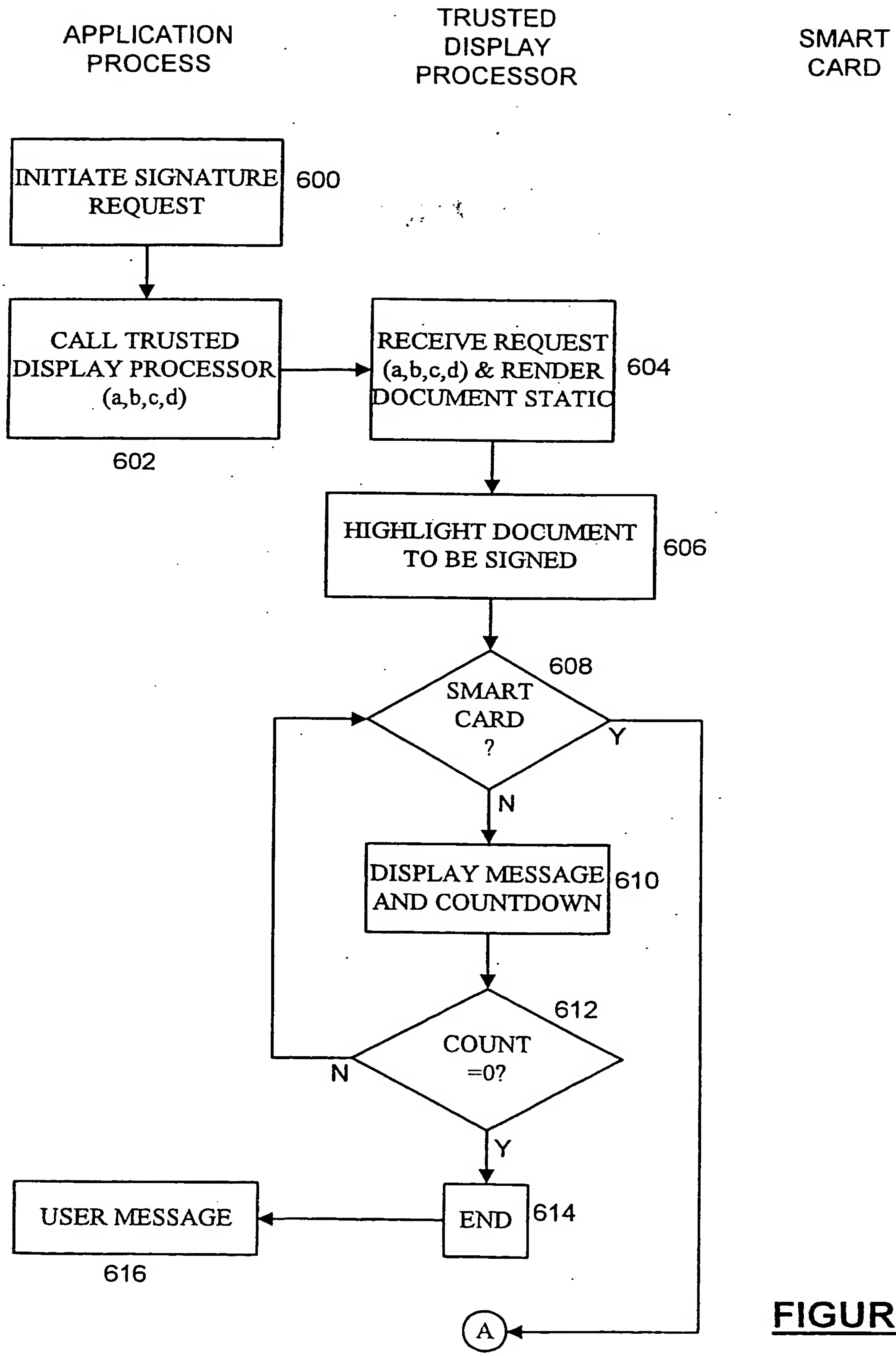

APPLICATION PROCESS
TRUSTED DISPLAY PROCESSOR
SMART CARD
INITIATE SIGNATURE REQUEST
600
CALL TRUSTED DISPLAY PROCESSOR (a,b,c,d)
602
RECEIVE REQUEST (a,b,c,d) & RENDER DOCUMENT STATIC
604
HIGHLIGHT DOCUMENT TO BE SIGNED
606
SMART CARD ?
608
Y
N
DISPLAY MESSAGE AND COUNTDOWN
610
COUNT =0?
612
N
Y
USER MESSAGE
616
END
614
A

FIGURE 6

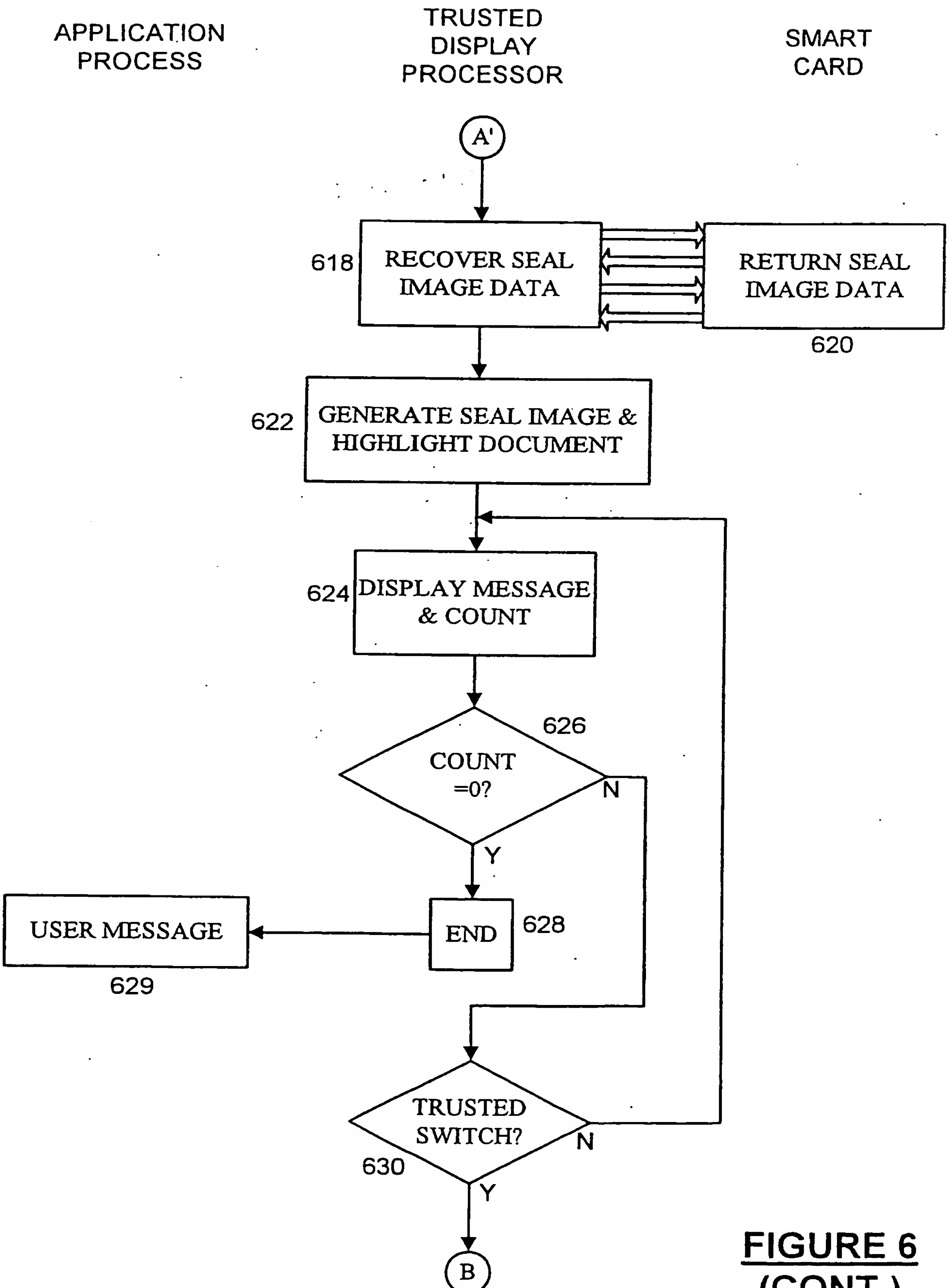


FIGURE 6 (CONT.)

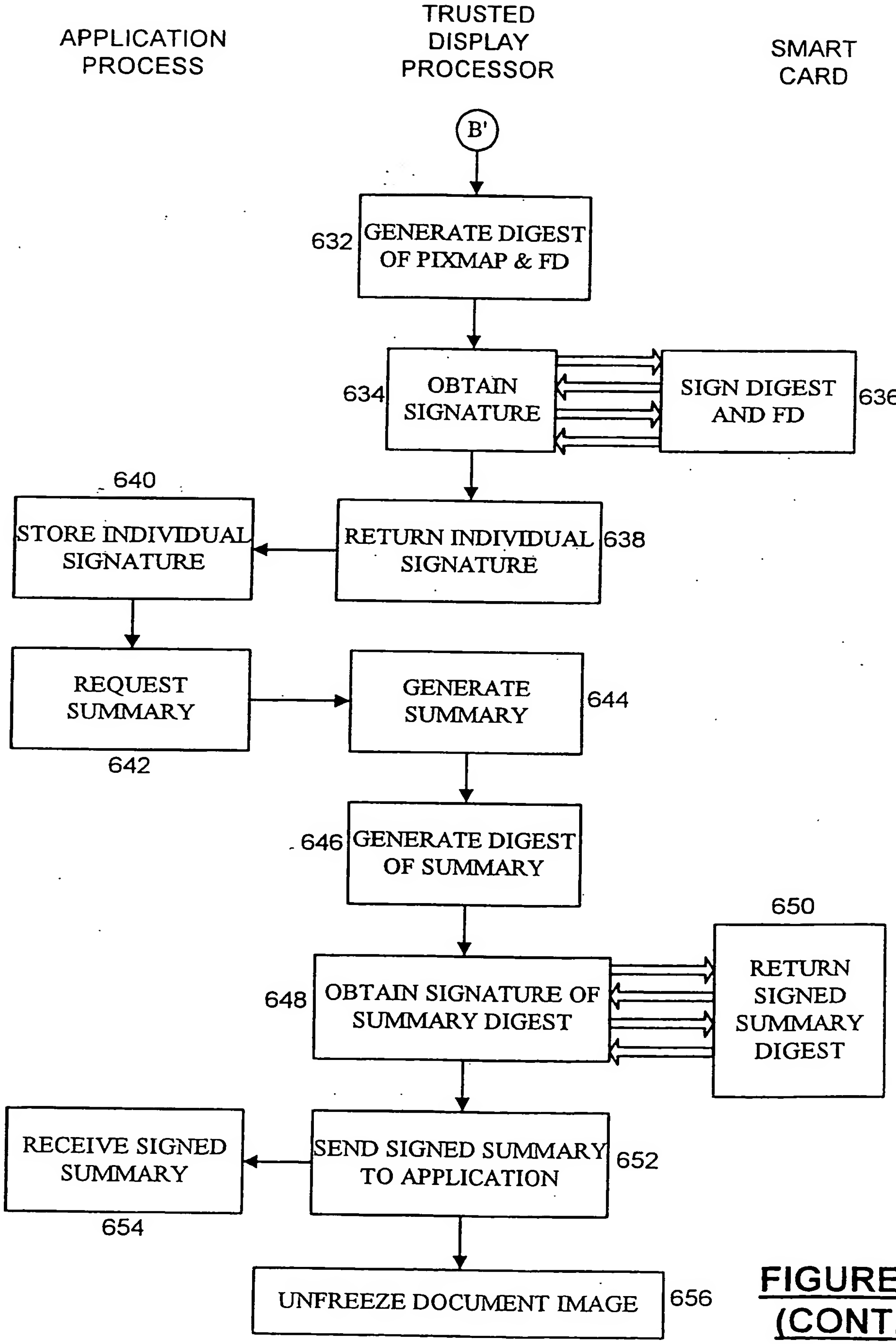


FIGURE 6 (CONT.)

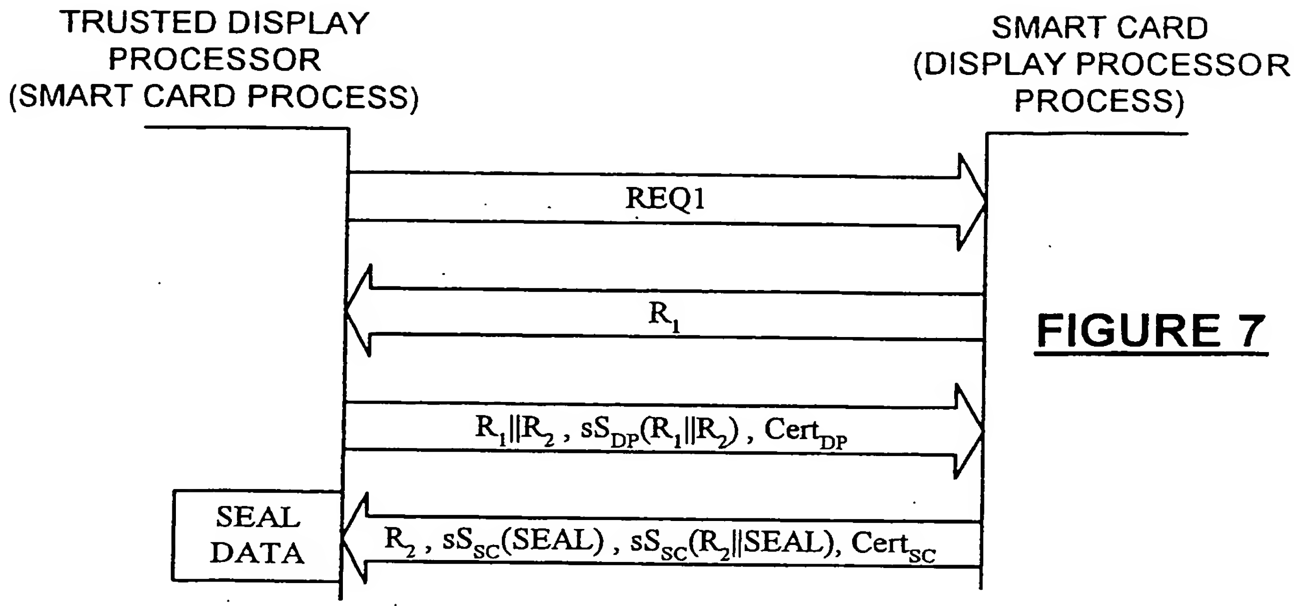


FIGURE 7

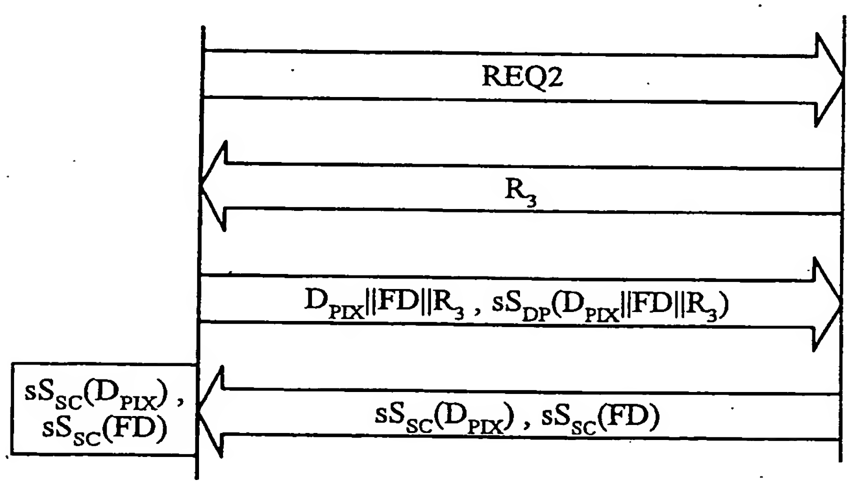


FIGURE 8

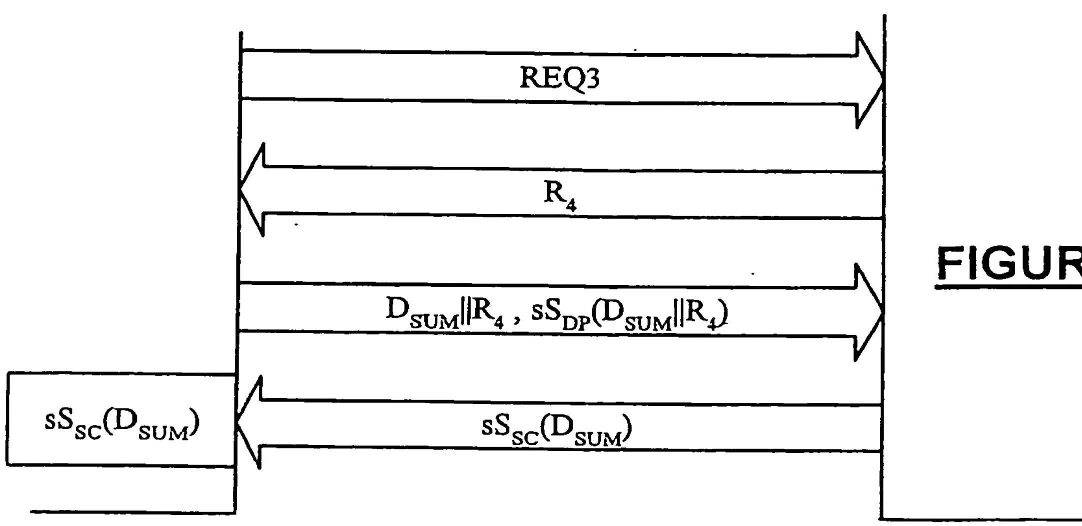


FIGURE 9

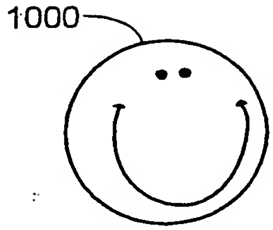


FIGURE 10a

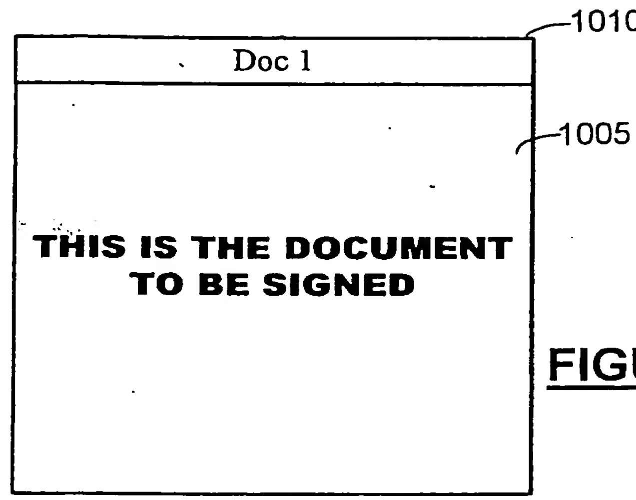


FIGURE 10b

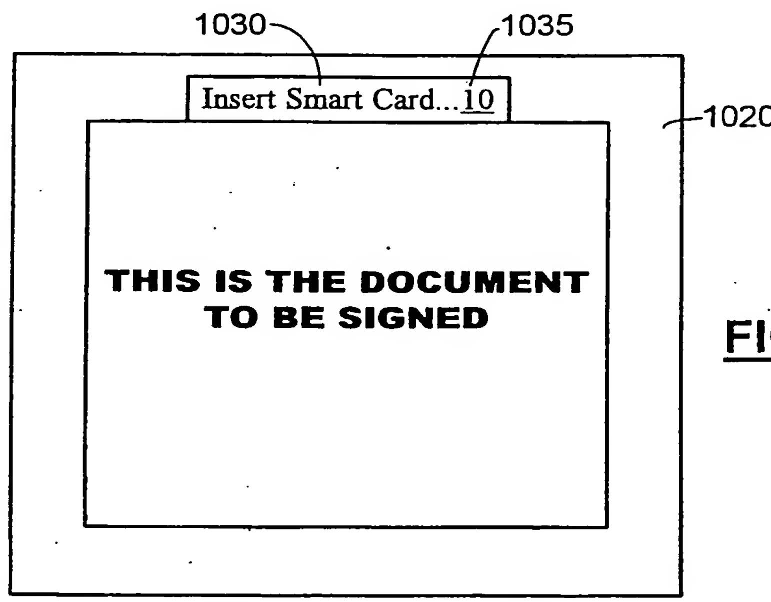


FIGURE 10c

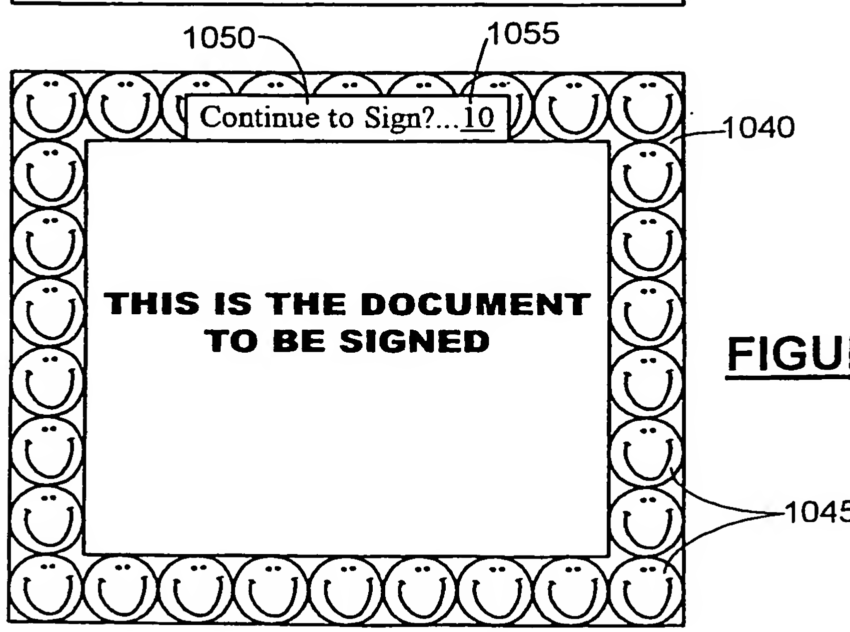


FIGURE 10d

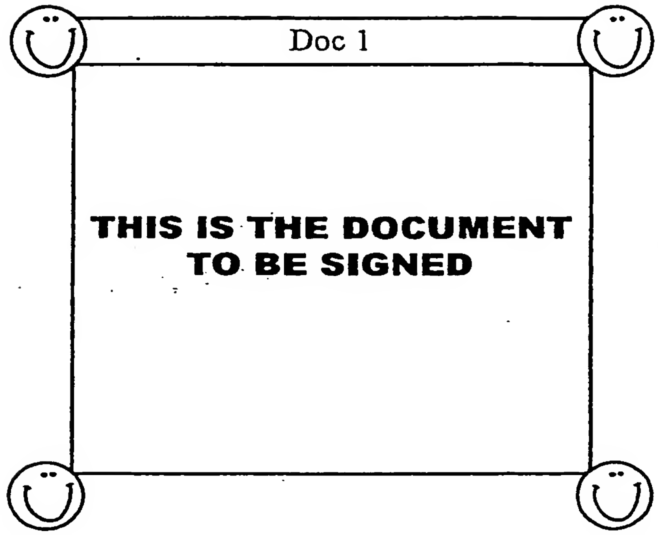


FIGURE 10e

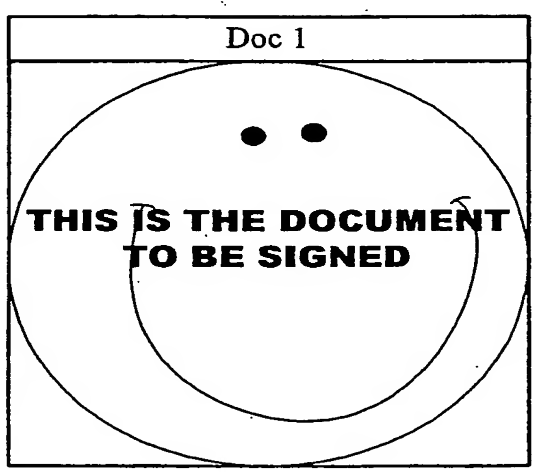


FIGURE 10f

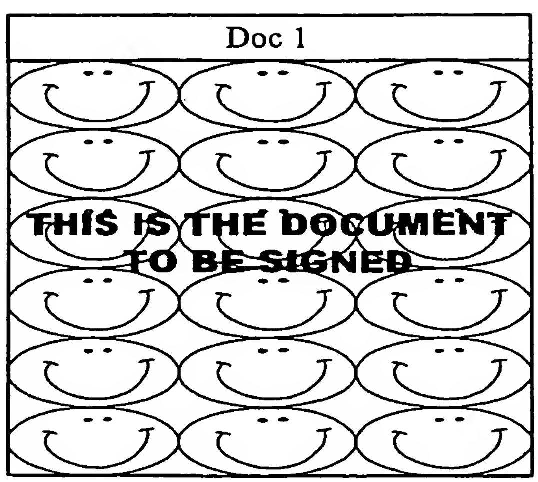


FIGURE 10g